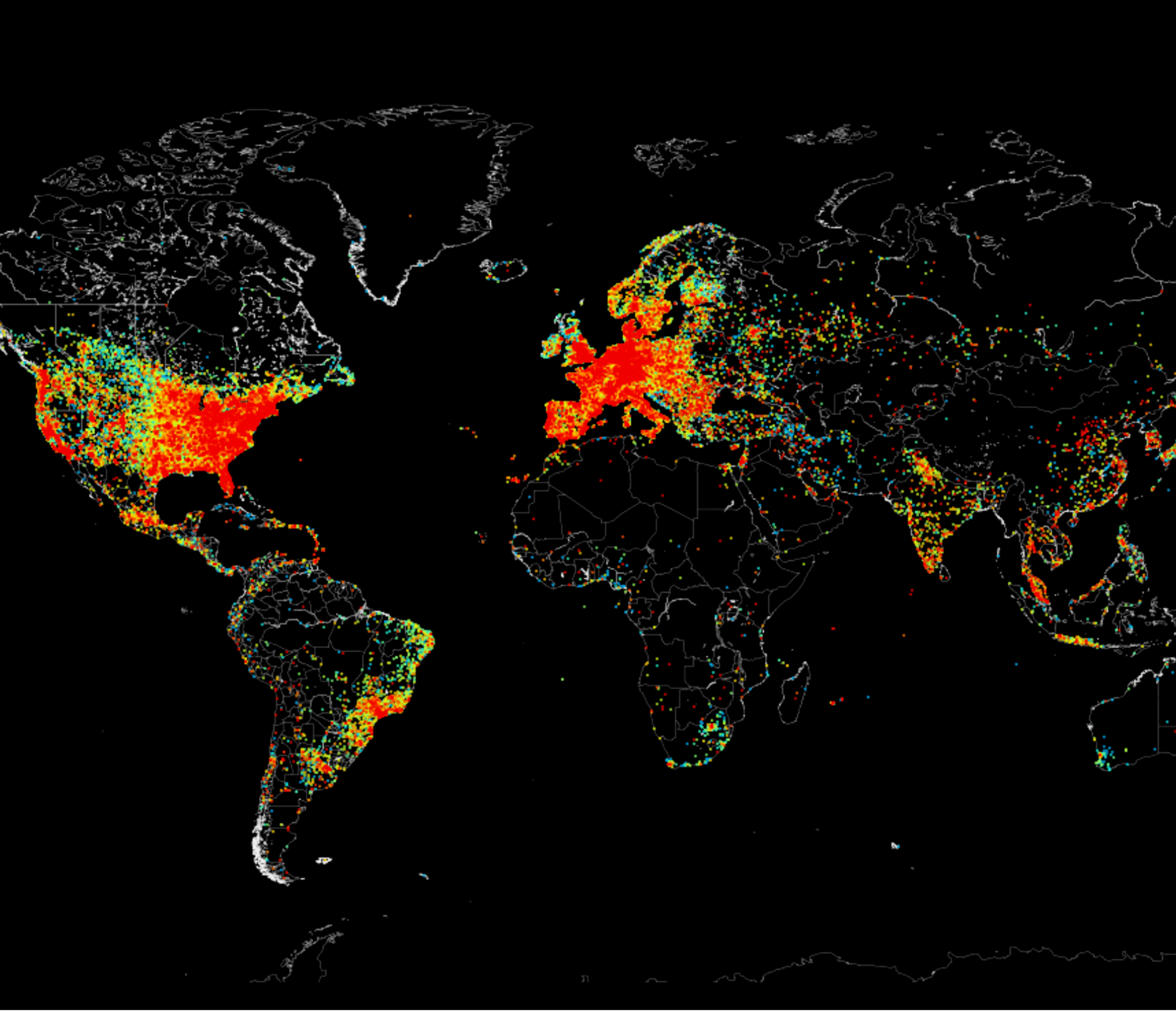


Complete Guide to Shodan

Collect. Analyze. Visualize. Make Internet Intelligence Work For You.

Complete Guide to Shodan

Collect. Analyze. Visualize. Make Internet Intelligence Work for You.

John Matherly

This book is for sale at http://leanpub.com/shodan

This version was published on 2016-10-10

Leanpub

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

© 2015 - 2016 Shodan, LLC

Tweet This Book!

Please help John Matherly by spreading the word about this book on Twitter!

The suggested hashtag for this book is #shodan.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

https://twitter.com/search?q=#shodan

Contents

Introduction

Shodan is a search engine for Internet-connected devices. Web search engines, such as Google and Bing, are great for finding websites. But what if you're interested in finding computers running a certain piece of software (such as Apache)? Or if you want to know which version of Microsoft IIS is the most popular? Or you want to see how many anonymous FTP servers there are? Maybe a new vulnerability came out and you want to see how many hosts it could infect? Traditional web search engines don't let you answer those questions.

All About the Data

Banner

The basic unit of data that Shodan gathers is the **banner**. The banner is textual information that describes a service on a device. For web servers this would be the headers that are returned or for Telnet it would be the login screen.

The content of the banner varies greatly depending on the type of service. For example, here is a typical HTTP banner:

```
HTTP/1.1 200 OK
Server: nginx/1.1.19
Date: Sat, 03 Oct 2015 06:09:24 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 6466
Connection: keep-alive
```

The above banner shows that the device is running the **nginx** web server software with a version of **1.1.19**. To show how different the banners can look like, here is a banner for the Siemens S7 industrial control system protocol:

```
Copyright: Original Siemens Equipment
PLC name: S7_Turbine
Module type: CPU 313C
Unknown (129): Boot Loader           A
Module: 6ES7 313-5BG04-0AB0  v.0.3
Basic Firmware: v.3.3.8
Module name: CPU 313C
Serial number of module: S Q-D9U083642013
Plant identification:
Basic Hardware: 6ES7 313-5BG04-0AB0  v.0.3
```

The Siemens S7 protocol returns a completely different banner, this time providing information about the firmware, its serial number and a lot of detailed data to describe the device.

You have to decide what type of service you're interested in when searching in Shodan because the banners vary greatly.

> **Note**: Shodan lets you search for banners - not hosts. This means that if a single IP exposes many services they would be represented as separate results.

Device Metadata

In addition to the banner, Shodan also grabs meta-data about the device such as its geographic location, hostname, operating system and more (see Appendix A). Most of the meta-data is searchable via the main Shodan website, however a few fields are only available to users of the developer API.

IPv6

As of October 2015, Shodan gathers millions of banners per month for devices accessible on IPv6. Those numbers still pale in comparison to the hundreds of millions of banners gathered for IPv4 but it is expected to grow over the coming years.

Data Collection

Frequency

The Shodan crawlers work 24/7 and update the database in real-time. At any moment you query the Shodan website you're getting the latest picture of the Internet.

Distributed

Crawlers are present in countries around the world, including:

- USA (East and West Coast)
- China
- Iceland
- France
- Taiwan
- Vietnam
- Romania
- Czech Republic

Data is collected from around the world to prevent geographic bias. For example, many system administrators in the USA block entire Chinese IP ranges. Distributing Shodan crawlers around the world ensures that any sort of country-wide blocking won't affect data gathering.

Randomized

The basic algorithm for the crawlers is:

1. Generate a random IPv4 address
2. Generate a random port to test from the list of ports that Shodan understands
3. Check the random IPv4 address on the random port and grab a banner
4. Goto 1

This means that the crawlers don't scan incremental network ranges. The crawling is performed completely random to ensure a uniform coverage of the Internet and prevent bias in the data at any given time.

SSL In Depth

SSL is becoming an evermore important aspect of serving and consuming content on the Internet, so it's only fit that Shodan extends the information that it gathers for every SSL-capable service. The banners for SSL services, such as HTTPS, include not just the SSL certificate but also much more. All the collected SSL information discussed below is stored in the **ssl** property on the banner (see Appendix A and Appendix E).

Vulnerability Testing

Heartbleed

If the service is vulnerable to Heartbleed then the banner contains 2 additional properties. **opts.heartbleed** contains the raw response from running the Heartbleed test against the service. Note that for the test the crawlers only grab a small overflow to confirm the service is affected by Heartbleed but it doesn't grab enough data to leak private keys. The crawlers also added **CVE-2014-0160** to the **opts.vulns** list if the device is vulnerable. However, if the device is not vulnerable then it adds **"!CVE-2014-0160"**. If an entry in **opts.vulns** is prefixed with a **!** or - then the service is **not vulnerable** to the given CVE.

```
{
    "opts": {
        "heartbleed": "... 174.142.92.126:8443 - VULNERABLE\n",
        "vulns": ["CVE-2014-0160"]
    }
}
```

Shodan also supports searching by the vulnerability information. For example, to search Shodan for devices in the USA that are affected by Heartbleed use:

```
country:US vuln:CVE-2014-0160
```

FREAK

If the service supports EXPORT ciphers then the crawlers add the "CVE-2015-0204" item to the **opts.vulns** property:

```
"opts": {
    "vulns": ["CVE-2015-0204"]
}
```

Logjam

The crawlers try to connect to the SSL service using ephemeral Diffie-Hellman ciphers and if the connection succeeds the following information is stored:

```
"dhparams": {
    "prime": "bbbc2dcad84674907c43fcf580e9...",
    "public_key": "49858e1f32aefe4af39b28f51c...",
    "bits": 1024,
    "generator": 2,
    "fingerprint": "nginx/Hardcoded 1024-bit prime"
}
```

Version

Normally, when a browser connects to an SSL service it will negotiate the SSL version and cipher that should be used with the server. They will then agree on a certain SSL version, such as TLSv1.2, and then use that for the communication.

Shodan crawlers start out the SSL testing by doing a normal request as outlined above where they negotiate with the server. However, afterwards they also explicitly try connecting to the server using a specific SSL version. In other words, the crawlers attempt to connect to the server using SSLv2, SSLV3, TLSv1.0, TLSv1.1 and TLSv1.2 explicitly to determine all the versions that the SSL service supports. The gathered information is made available in the **ssl.versions** field:

```
{
    "ssl": {
        "versions": ["TLSv1", "SSLv3", "-SSLv2", "-TLSv1.1", "-TLSv1.2"]
    }
}
```

If the version has a - (dash) in front of the version, then the device does not support that SSL version. If the version doesn't begin with a -, then the service supports the given SSL version. For example, the above server supports:

```
TLSv1
SSLv3
```

And it denies versions:

```
SSLv2
TLSv1.1
TLSv1.2
```

The version information can also be searched over the website/ API. For example, the following search query would return all SSL services (HTTPS, POP3 with SSL, etc.) that allow connections using SSLv2:

```
ssl.version:sslv2
```

Follow the Chain

The certificate chain is the list of SSL certificates from the root to the end-user. The banner for SSL services includes a **ssl.chain** property that includes all of the SSL certificates of the chain in PEM-serialized certificates.

Beyond the Basics

For most services the crawlers attempt to analyze the main banner text and parse out any useful information. A few examples are the grabbing of collection names for MongoDB, taking screenshots from remote desktop services and storing a list of peers for Bitcoin. There are 2 advanced data analysis techniques Shodan uses that I'd like to highlight:

Web Components

The crawlers try to determine the web technologies that were used to create a website. For the **http** and **https** modules the headers and HTML are analyzed to breakdown the components of the website. The resulting information is stored in the **http.components** property. The property is a dictionary of technologies, where the key is the name of the technology (ex. jQuery) and the value is another dictionary with a property of categories. The categories property is a list of categories that are associated with the technology. For example:

```
"http": {
...
    "components": {
        "jQuery": {
            "categories": ["javascript-frameworks"]
        },
        "Drupal": {
            "categories": ["cms"]
        },
        "PHP": {
            "categories": ["programming-languages"]
        }
    },
    ...
},
```

The **http.components** property indicates that the website is running the Drupal content management system, which itself uses jQuery and PHP. The Shodan REST API makes the information searchable via the filter **http.component** and 2 facets (**http.component** and **http.component_category**). To get a full list of all the possible component/ category values please use the new facets. For example, to get a full list of all the possible categories use the following shodan command:

```
$ shodan stats --facets http.component_category:1000 http
    Top 47 Results for Facet: http.component_category
    javascript-frameworks          8,982,996
    web-frameworks                 1,708,503
    programming-languages          1,409,763
    font-scripts                   1,280,397
```

Cascading

If a banner returns information about peers or otherwise has information about another IP address that runs a service then the crawlers try to perform a banner grab on that IP/ service. For example: the default port for the mainline DHT (used by Bittorrent) is 6881. The banner for such a DHT node looks as follows:

```
DHT Nodes
97.94.250.250        58431
150.77.37.22        34149
113.181.97.227        63579
252.246.184.180        36408
83.145.107.53        52158
77.232.167.126        52716
25.89.240.146        27179
147.23.120.228        50074
85.58.200.213        27422
180.214.174.82        36937
241.241.187.233        60339
166.219.60.135        3297
149.56.67.21        13735
107.55.196.179        8748
```

Previously, a crawler would grab the above banner and then move on. With cascading enabled for the DHT banner grabber the crawler now launches new banner grabbing requests for all of the peers. In the above example, the crawler would launch a scan for IP 54.70.96.157 on port 61770 using the dht banner grabber, IP 85.82.92.188 on port 42155 and so on. I.e. a single scan for an IP can cause a cascade of scans if the initial scan data contains information about other potential hosts.

To keep track of the relationship between the initial scan request and any child/ cascading requests we've introduced 2 new properties:

- **_shodan.id**: A unique ID for the banner. This property is guaranteed to exist if a cascading request could get launched from the service, though it doesn't necessarily mean that any cascading requests succeeded.
- **_shodan.options.referrer**: Provides the unique ID of the banner that triggered the creation of the current banner. I.e. the referrer is the parent of the current banner.

Web Interfaces

The easiest way to access the data that Shodan gathers is through the web interfaces. Almost all of them let you enter a search query, so lets discuss that first:

Search Query Explained

By default, the search query only looks at the main banner text and doesn't search the meta-data. For example, if you're searching for "Google" then the results will only include results where the text "Google" was shown in the banner; it wouldn't necessarily return results for Google's network range.

Shodan search for "Google"

As seen above, a search for "Google" returns a lot of Google Search Appliances that organizations have purchased and connected to the Internet; it doesn't return Google's servers.

Shodan will try to find results matching **all search terms**, which means that implicitly there is a **+** or **AND** between each search term. For example, the search queries "apache + 1.3" is equivalent to "apache 1.3".

To search the meta-data you need to use **search filters**.

Introducing Filters

Filters are special keywords that Shodan uses to let you narrow search results based on the meta-data of a service or device. The format for entering filters is:

```
filtername:value
```

> **Important**: There is no space between the colon ":" and the value.

To use a value that contains a space with a filter you have to wrap the value in double quotes. For example, to find all devices on the Internet that are located in San Diego you would search for:

```
city:"San Diego"
```

A few filters let you specify several values that are separated by a comma ",". For example, to find devices that are running Telnet on ports 23 and 1023:

```
port:23,1023
```

If a filter doesn't allow commas in its value (ex. **port**, **hostname**, **net**) then it lets you provide multiple values.

Filters can also be used to exclude results by prepending a minus sign "-" to the filter. For example, the following would return all devices that **aren't** located in San Diego:

```
-city:"San Diego"
```

There are many situations where excluding is easier than including. For example, the following search query uses **hash:0** to provide results for services on port 8080 where the main text banner isn't empty:

```
port:8080 -hash:0
```

Every banner on Shodan has a numeric **hash** property calculated; for empty banners that value is zero. If you're trying to find devices that have a short, static banner then the **hash** filter may provide a good way to accurately identify them.

Shodan supports a lot of filters, a few popular ones are:

Filter Name	Description	Example
category	Available categories: ics, malware	
city	Name of the city	
country	Full country name	
net	Only show results inside the provided IP range in CIDR format	net:190.30.40.0/24
org	Narrow results based on the organization that owns the IP	org:"Verizon Wireless"

See **Appendix B** for a full list of search filters that are available.

Shodan Search Engine

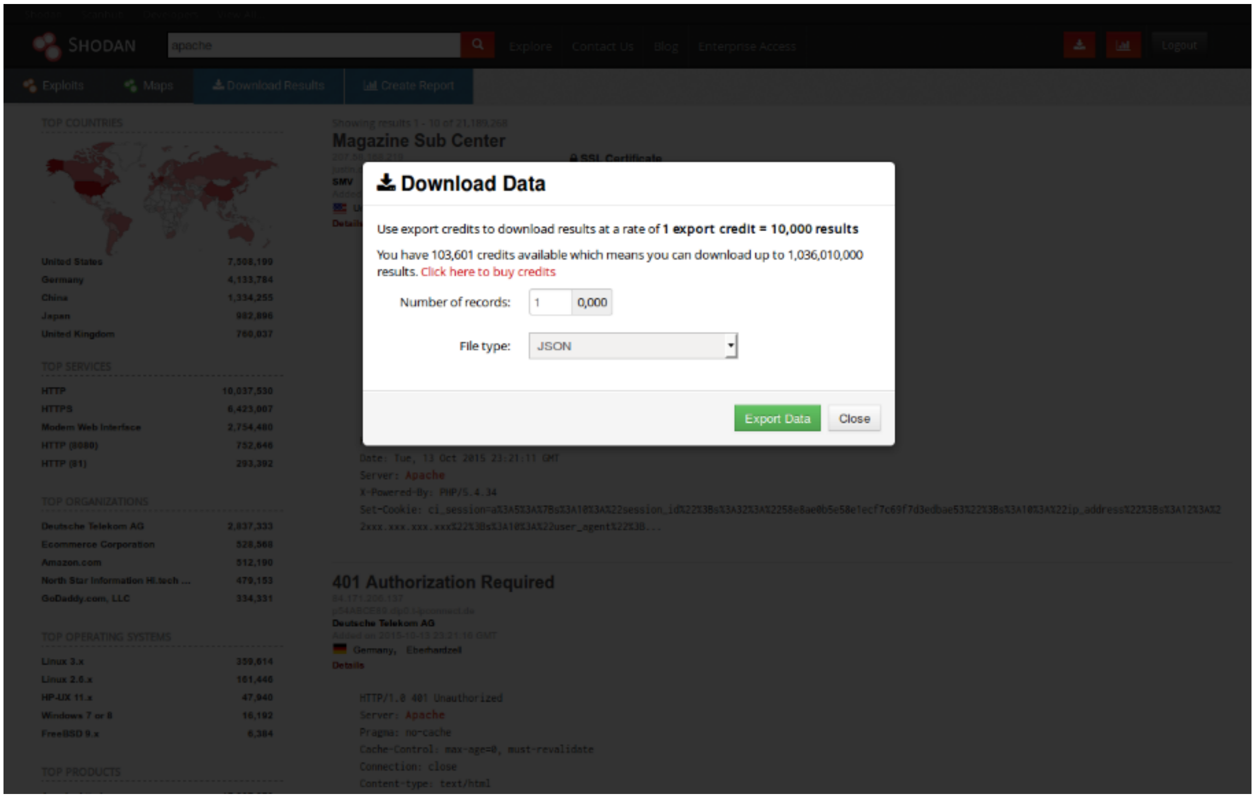
The main interface for accessing the data gathered by Shodan is via its search engine located at https://www.shodan.io

By default, the search query will look at the data collected within the past 30 days. This is a change from the old website at shodanhq.com, which searched the entire Shodan database by default. This means that the results you get from the website are recent and provide an accurate view of the Internet at the moment.

In addition to searching, the website also provides the following functionality:

Download Data

After completing a search there will be a button at the top called **Download Data**. Clicking on that button will provide you with the option of downloading the search results in **JSON**, **CSV** or **XML** formats.

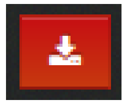


The **JSON** format generates a file where each line contains the full banner and all accompanying meta-data that Shodan gathers. This is the preferred format as it saves all available information. And the format is compatible with the Shodan command-line client, meaning you can download data from the Shodan website then process it further using the terminal.

The **CSV** format returns a file containing the IP, port, banner, organization and hostnames for the banner. It doesn't contain all the information that Shodan gathers due to limitations in the CSV file format. Use this if you only care about the basic information of the results and want to quickly load it into external tools such as Excel.

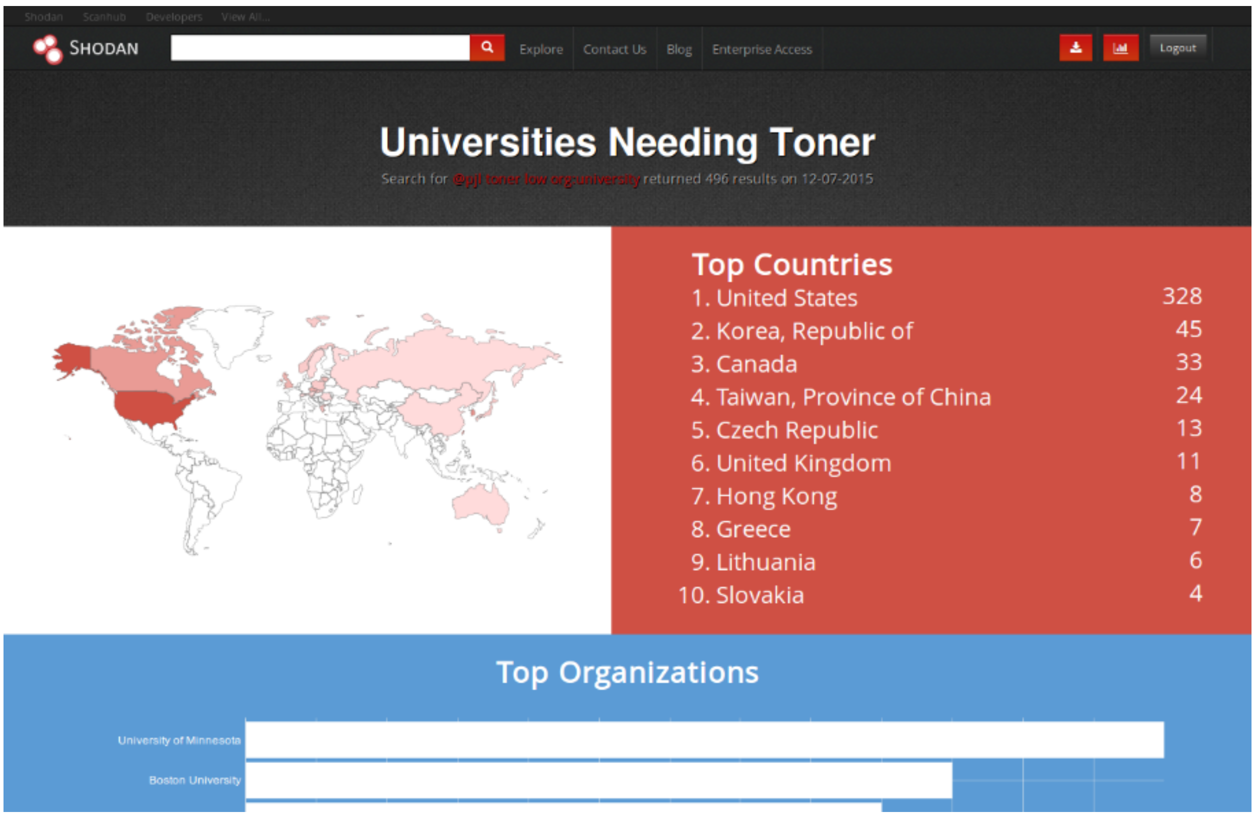
The **XML** format is the old, deprecated way of saving search results. It is harder to work with than JSON and consumes more space, thereby making it suboptimal for most situations.

Downloading data consumes **export credits**, which are one-time use and purchased on the website. They aren't associated in any way with the Shodan API and they don't automatically renew every month. 1 export credit can be used to download up to 10,000 results.

Data files generated by the website can be retrieved in the Downloads section of the website, which you can visit by clicking on the  button in the upper right corner.

Generate Report

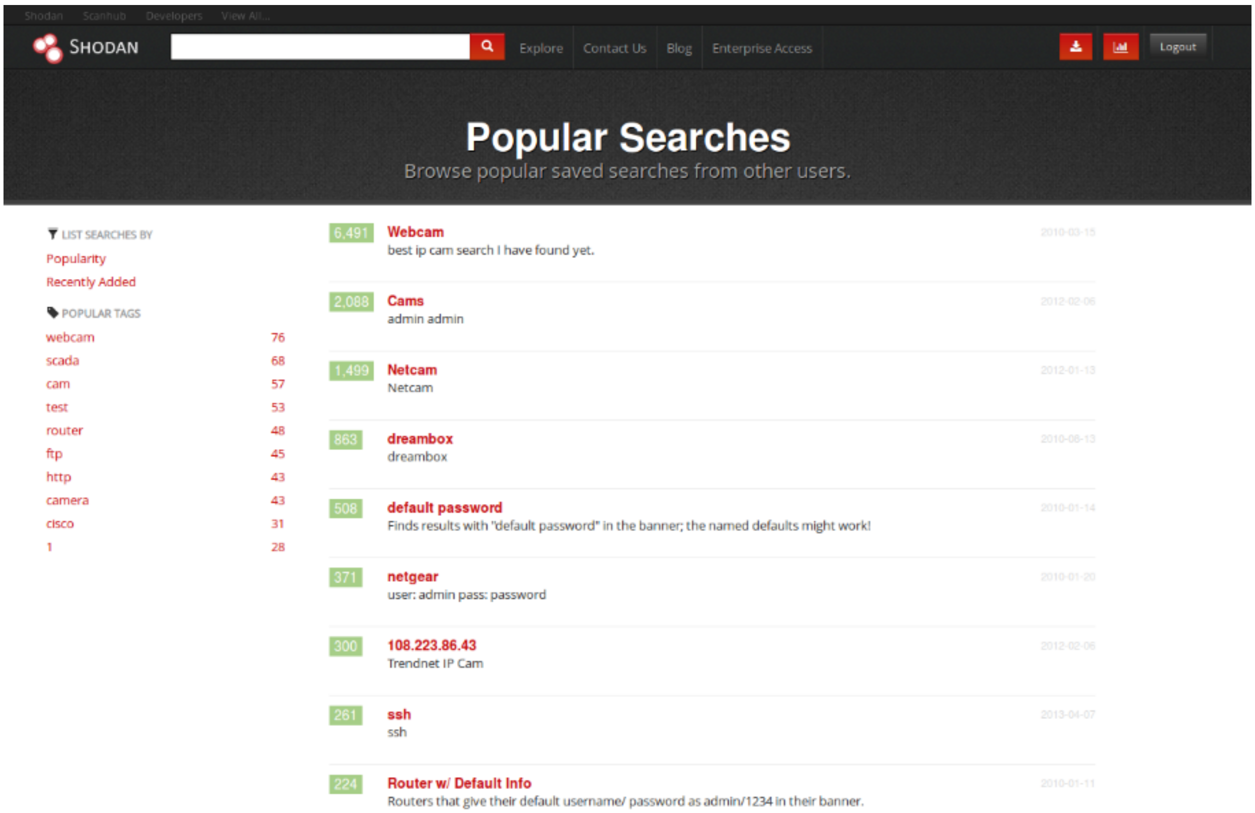
The website lets you generate a report based off of a search query. The report contains graphs/ charts providing you a big picture view of how the results are distributed across the Internet. This feature is free and available to anyone.



When you generate a report you are asking Shodan to take a snapshot of the search results and provide an aggregate overview. Once the report has been generated, it doesn't change or automatically update as new data is being collected by Shodan. This also means that you can generate a report once a month and keep track of changes over time by comparing it to reports of previous months. By clicking on the  button in the top right corner you can get a listing of previously generated reports.

Shared Search Queries

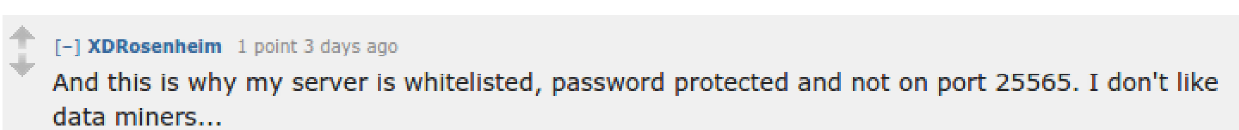
Finding specific devices requires knowledge about the software they run and how they respond to banner grabs over the Internet. Fortunately, it is possible to leverage the shared knowledge of the community using the search directory on Shodan. People are able to readily describe, tag and share their search queries for others to use. If you're interested in getting started with Shodan, the shared searches should be your first stop.



> **Warning**: Shared search queries are publicly viewable. Do not share queries that are sensitive or you don't want others to know about.

Example: Finding Non-Default Services

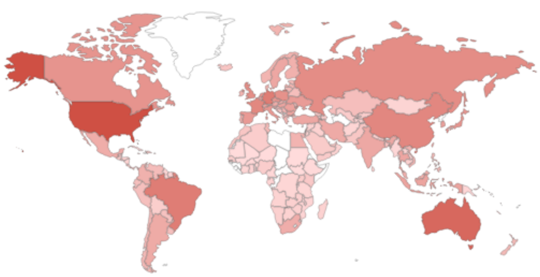
A common reaction I get when talking about devices exposed on the Internet is something like the following:



Specifically, the idea that running the service (in this case Minecraft) on a non-standard port is a good way to stay hidden. In security circles this is also known as the concept of security by obscurity, and it's considered a largely ineffective, deprecated idea. What's worse is that it might give you the owner of the server/ device a false sense of security. For example, lets take a look at people running OpenSSH on a non-standard port. To do this we will use the following search query:

```
product:openssh -port:22
```

The **product** filter is used to only show OpenSSH servers while **-port:22** tells Shodan to exclude all results that were collected from the standard SSH port (22). To get a better overview of the search results lets generate a report:



Top Countries

1.	United States	139,969
2.	Australia	59,493
3.	Germany	24,584
4.	Brazil	24,405
5.	China	15,123
6.	France	14,708
7.	Russian Federation	11,065
8.	United Kingdom	10,692
9.	Poland	8,496
10.	Canada	7,484

The report also gives us a breakdown of the most common non-standard ports:

1. **2222**: 323,930
2. **5000**: 47,439
3. **23**: 13,482
4. **26**: 7,569
5. **5555**: 6,856
6. **9999**: 6,286
7. **82**: 6,046
8. **2323**: 3,622
9. **6666**: 2,735
10. **3333**: 2,644

These numbers don't look that random to me… Right away you should realize that your random choice of non-standard port might not be so unique. Port 2222 is popular the same way that HTTP on port 8080 is popular, and it's also the default port for the Kippo honeypot though I doubt that many people are running honeypots. The next most popular port is 5000, which didn't follow the same pattern as the other ports to me (repeating/ symmetric numbers). And it was around the same time that I realized that Australia was the 2nd most popular country to run OpenSSH on a non-standard port. I decided to take a closer look at Australia, and it turns out that there are nearly the same amount of servers running OpenSSH on port 5000 as they are on the default port 22. About 68,000 devices are running on the default port, and 54,000 on port 5000. Looking at a few banners we can determine that this is the SSH fingerprint that they all share:

```
5b:a2:5a:9a:91:28:60:9c:92:2b:9e:bb:7f:7c:2e:06
```

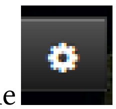
It appears that the Australian ISP BigPond installs/ configures networking gear that not only runs OpenSSH on port 5000 (most likely for remote management) but also has the same SSH keys installed on all of them. The devices also happen to run an old version of OpenSSH that was released on September 4th 2007. There's no guarantee that running OpenSSH on the default port would've made them more security conscious, but their installation of ∼54,000 devices is 25% of the total number of OpenSSH servers on the Internet running version 4.7 (sidenote: the most popular version of OpenSSH is 5.3).

Shodan Maps

Shodan Maps[1] provides a way to explore search results visually instead of the text-based main website. It displays up to 1,000 results at a time and as you zoom in/ out Maps adjusts the search query to only show results for the area you're looking at.

All search filters that work for the main Shodan website also work on Maps.

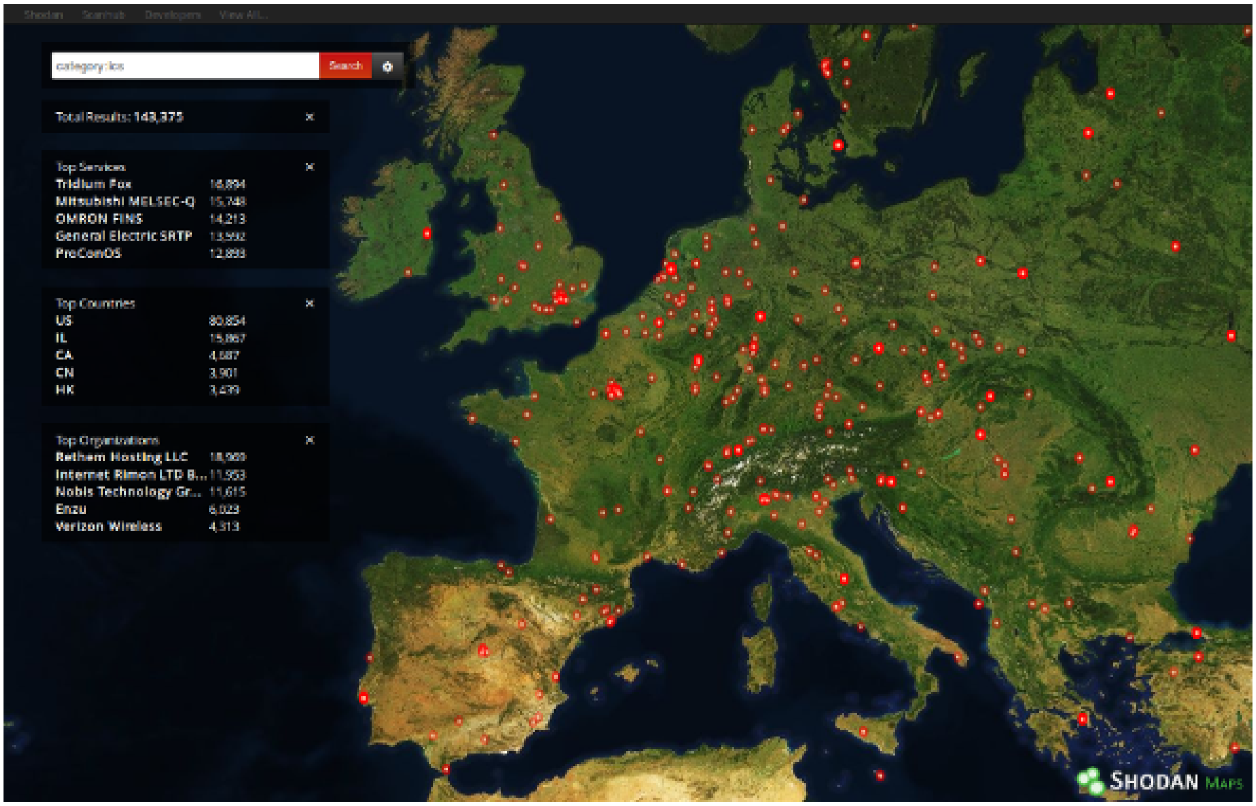
Map Styles

There are a variety of map styles available to present the data to your preference. Click on the  gear button next to the search button for a list of options.

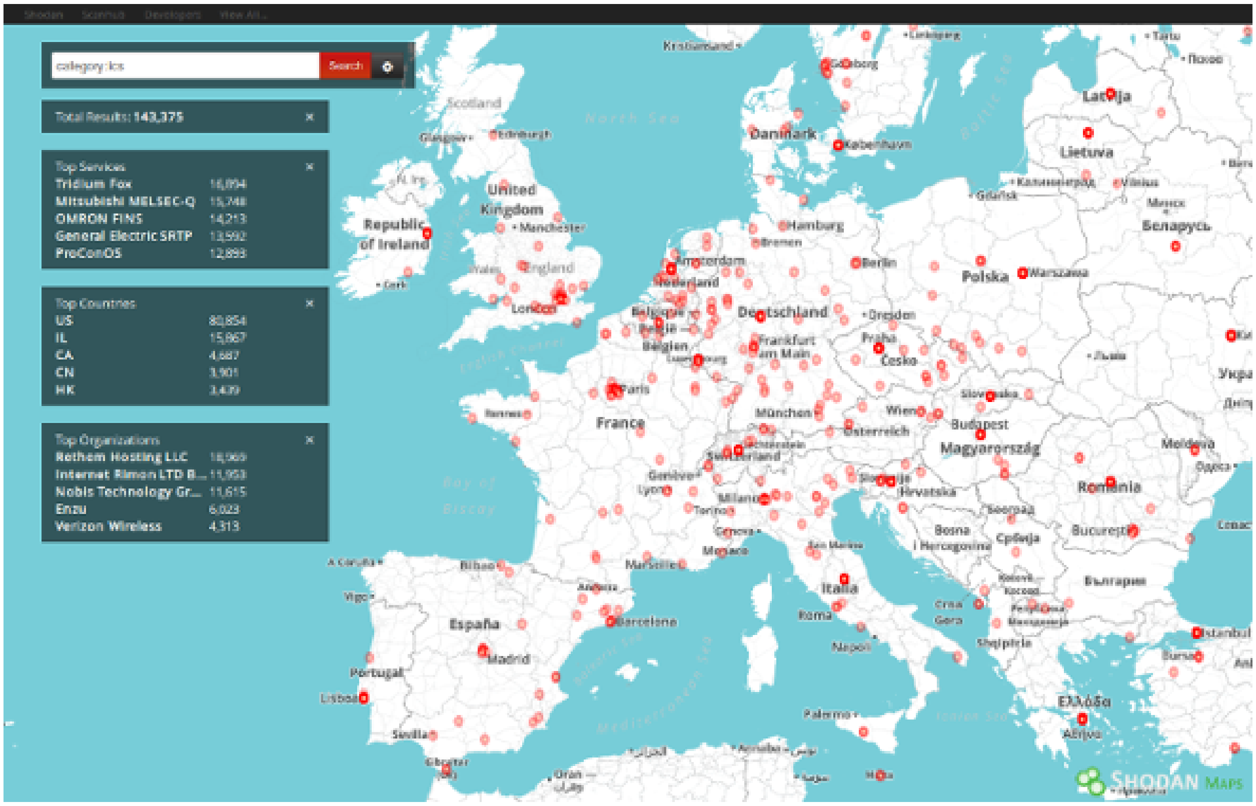
Satellite

[1] https://maps.shodan.io

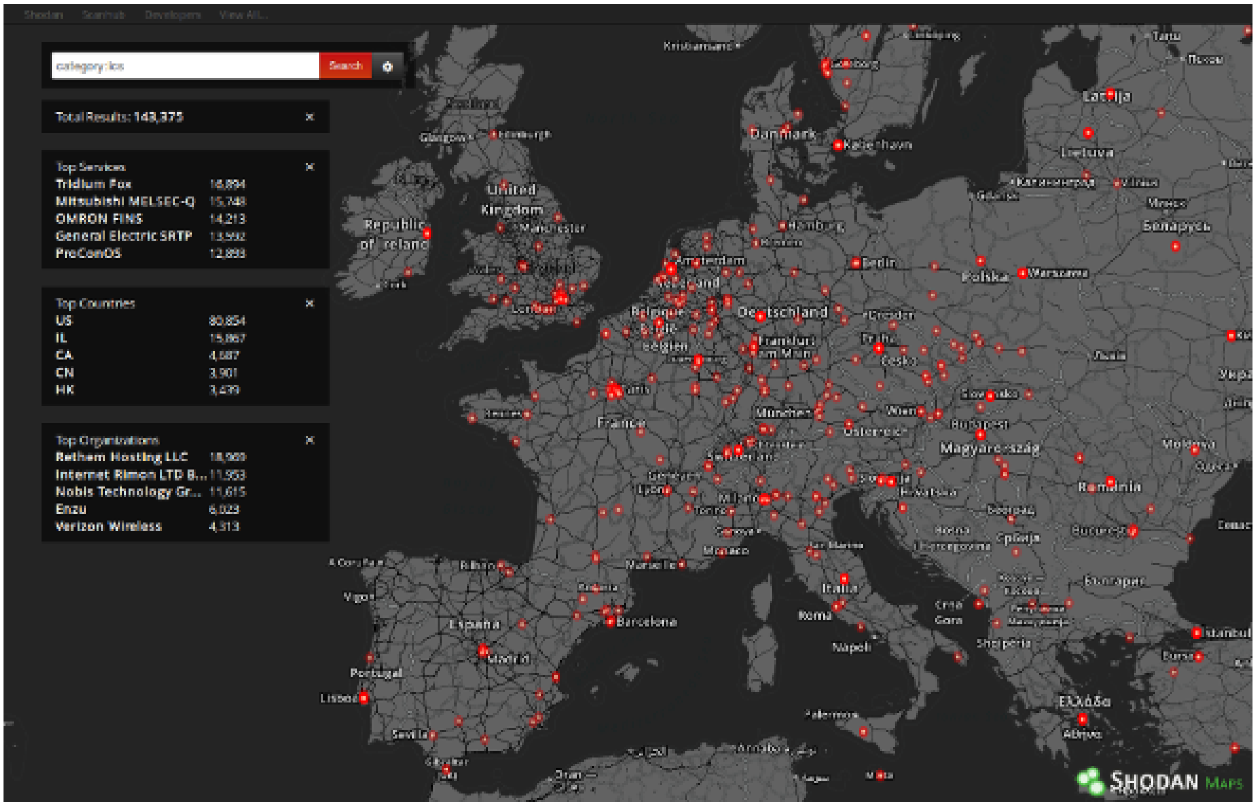
Satellite without Labels



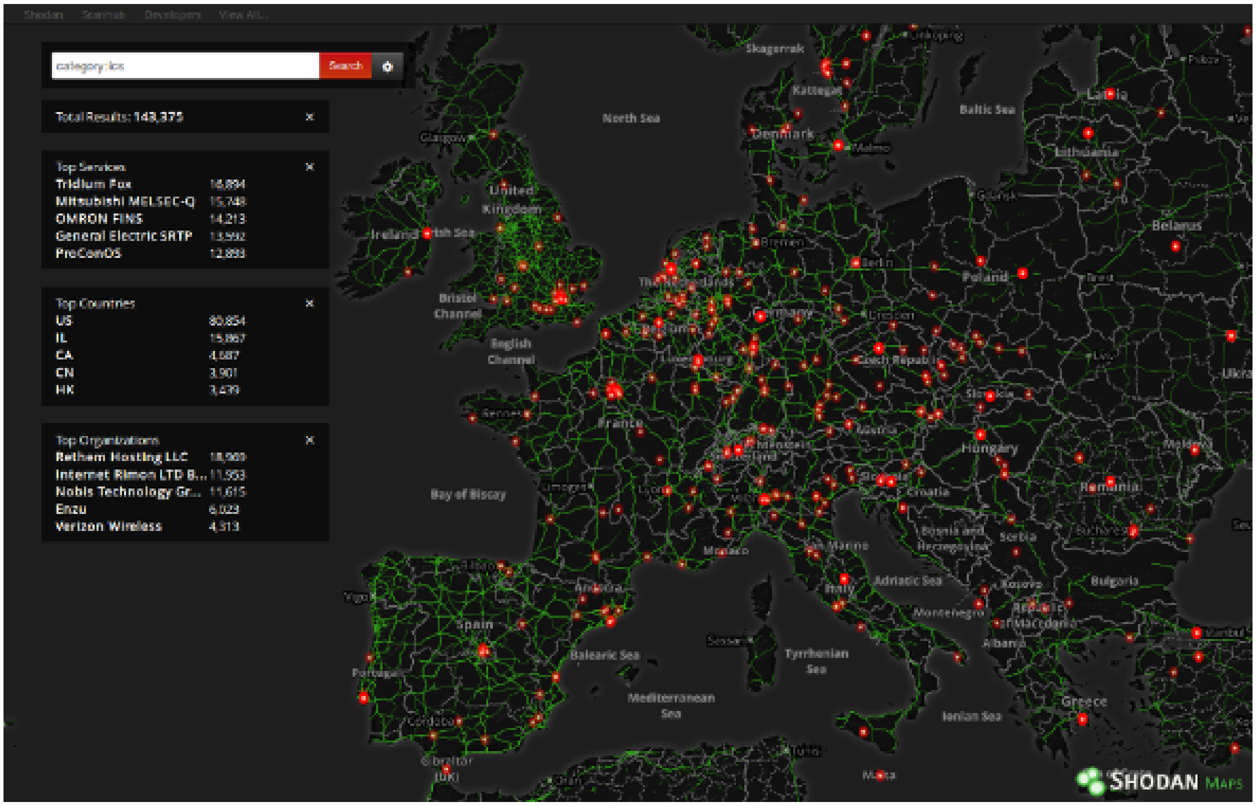
Streets (Light)



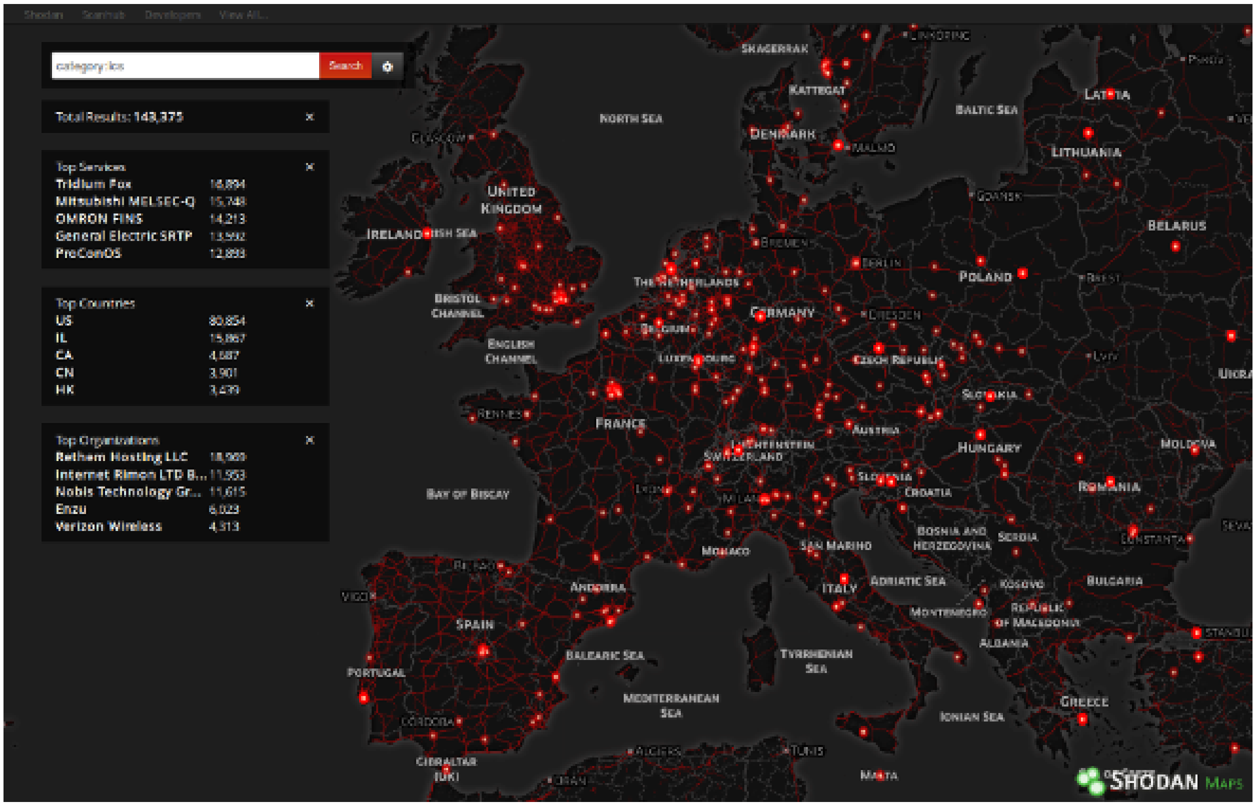
Streets (Dark)



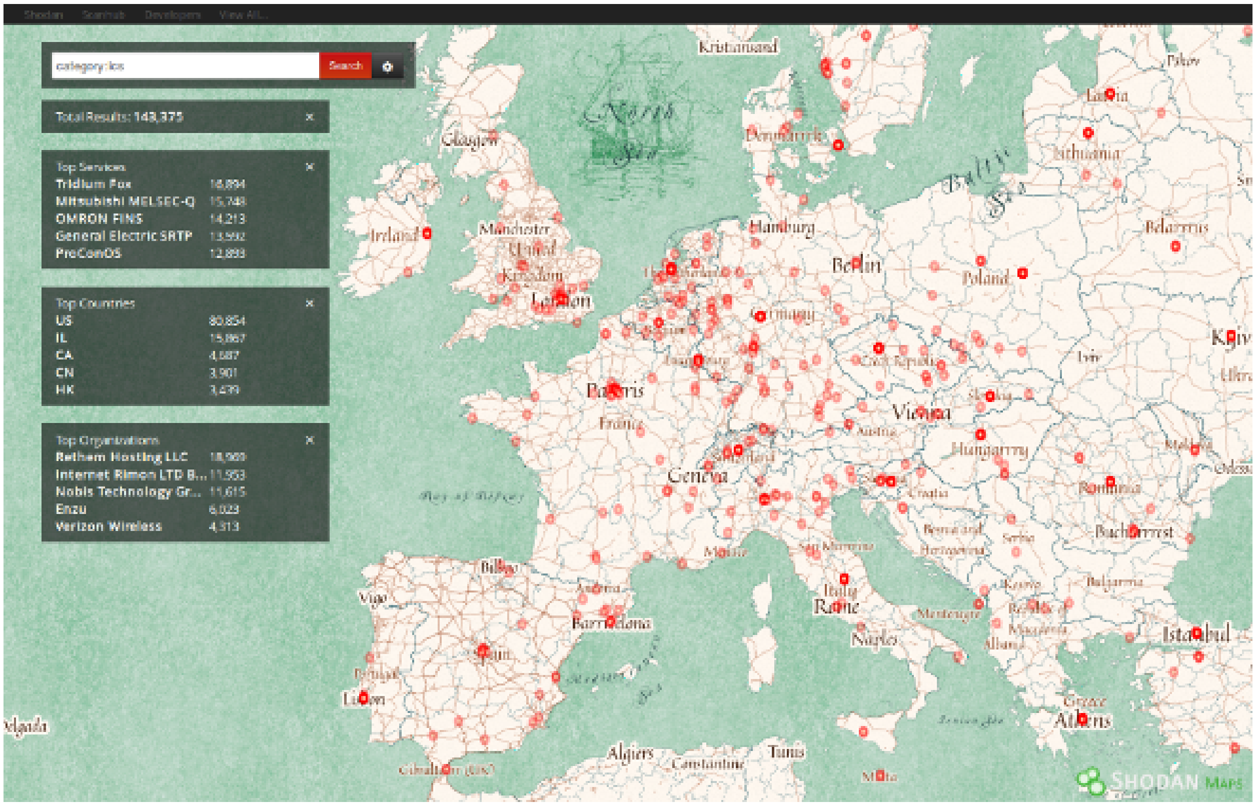
Streets (Green)



Streets (Red)

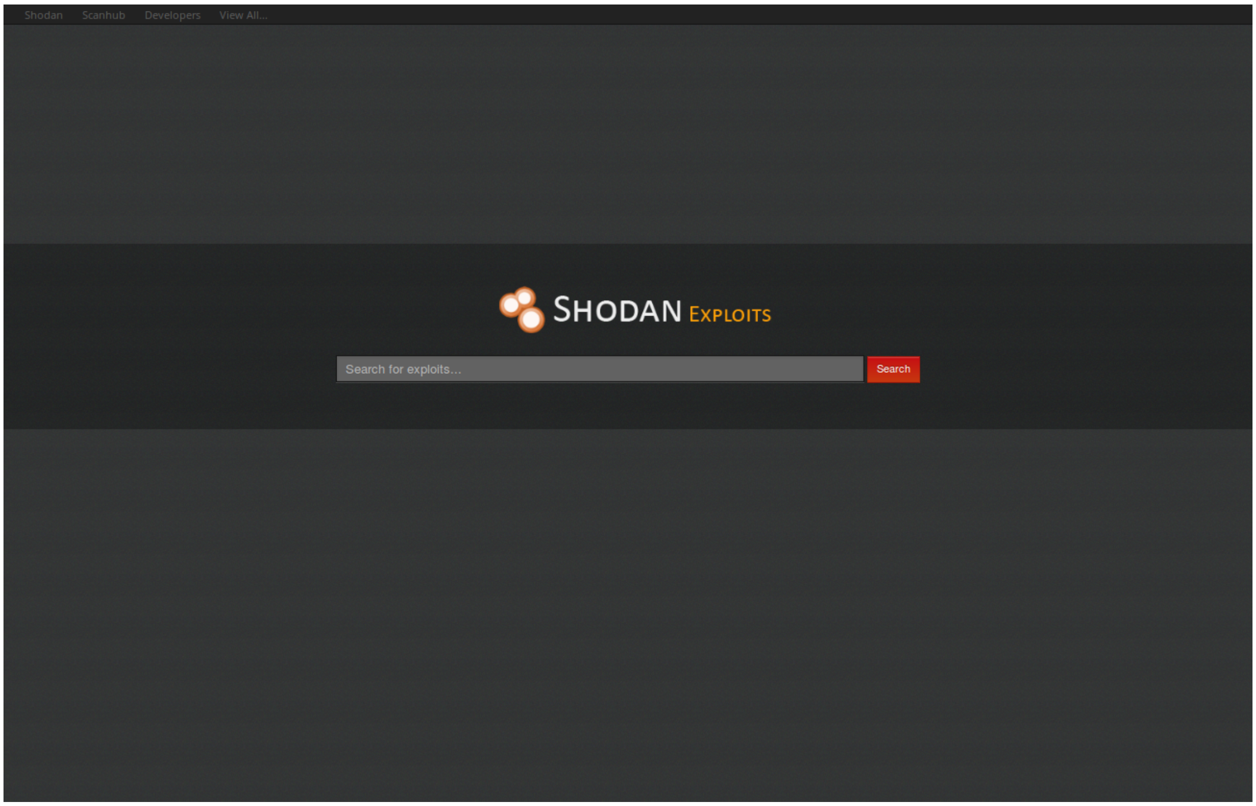

Shodan
Scanhub
Developers
View All...
category:ics
Search
Total Results: 143,375
Top Services
Tridium Fox 16,894
Mitsubishi MELSEC-Q 15,748
OMRON FINS 14,213
General Electric SRTP 13,592
ProConOS 12,893
Top Countries
US 80,854
IL 15,867
CA 4,687
CN 3,901
HK 3,439
Top Organizations
Rethem Hosting LLC 18,969
Internet Rimon LTD B... 11,953
Nobis Technology Gr... 11,615
Enzu 6,023
Verizon Wireless 4,313
NORTH SEA
UNITED KINGDOM
IRELAND
DENMARK
BALTIC SEA
LATVIA
LITHUANIA
BELARUS
POLAND
GERMANY
THE NETHERLANDS
BELGIUM
LUXEMBOURG
CZECH REPUBLIC
SLOVAKIA
AUSTRIA
HUNGARY
MOLDOVA
ROMANIA
FRANCE
SWITZERLAND
LIECHTENSTEIN
SLOVENIA
CROATIA
BOSNIA AND HERZEGOVINA
SERBIA
KOSOVO
BULGARIA
MONTENEGRO
ALBANIA
GREECE
ITALY
SAN MARINO
MONACO
ANDORRA
SPAIN
PORTUGAL
BAY OF BISCAY
ENGLISH CHANNEL
BRISTOL CHANNEL
ADRIATIC SEA
TYRRHENIAN SEA
IONIAN SEA
MEDITERRANEAN SEA
BALEARIC SEA
MALTA
SKAGERRAK
KATTEGAT
SHODAN MAPS

Pirate


Shodan
Scanhub
Developers
View All...
category:ics
Search
Total Results: 143,375
Top Services
Tridium Fox 16,894
Mitsubishi MELSEC-Q 15,748
OMRON FINS 14,213
General Electric SRTP 13,992
ProConOS 12,893
Top Countries
US 80,854
IL 15,867
CA 4,687
CN 3,901
HK 3,479
Top Organizations
Rethem Hosting LLC 18,989
Internet Rimon LTD B... 11,953
Nobis Technology Gr... 11,615
Enzu 6,023
Verizon Wireless 4,313
SHODAN MAPS

Shodan Exploits

Shodan Exploits[2] collects vulnerabilities and exploits from CVE, Exploit DB and Metasploit to make it searchable via web interface.



The search filters available for Exploits are different than the rest of Shodan, though an attempt was made to keep them similar when possible.

> **Important**: By default, Exploits will search the entire content of the available exploit information including meta-data. This is unlike Shodan, which only searches the banner text if no other filters are specified.

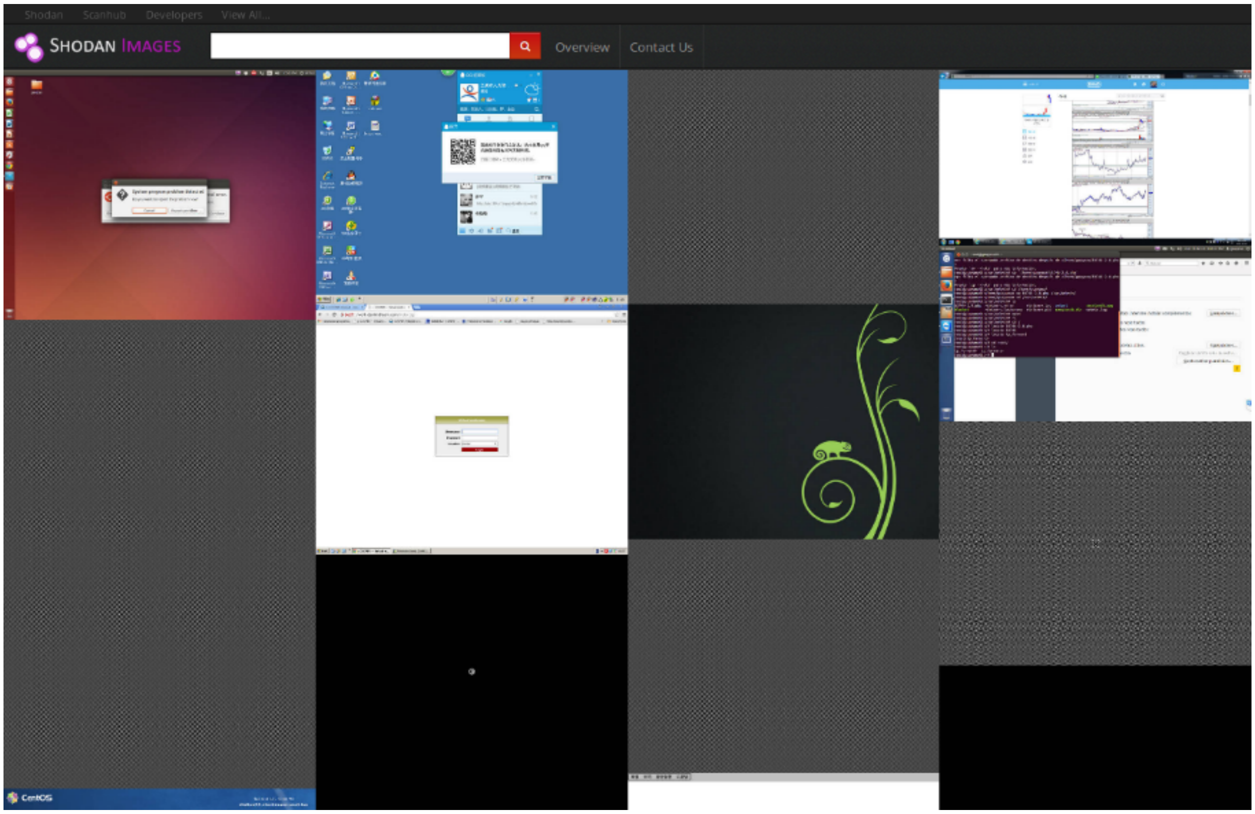
The following search filters are available:

Name	Description
author	Author of the vulnerability/ exploit
description	Description
platform	Platform that it targets (ex: php, windows, linux)
type	Exploit type (ex: remote, dos)

[2] https://exploits.shodan.io

Shodan Images

For a quick way to browse all the screenshots that Shodan collects check out Shodan Images[3]. It is a user-friendly interface around the **has_screenshot** filter.



The search box at the top uses the same syntax as the main Shodan search engine. It is most useful to use the search box to filter by organization or netblock. However, it can also be used to filter the types of images that are shown.

Image data is gathered from 5 different sources:

- VNC
- Remote Desktop (RDP)
- RTSP
- Webcams
- X Windows

Each image source comes from a different port/ service and therefor has a different banner. This means that if you only want to see images from webcams you could search for[4]:

[3] https://images.shodan.io
[4] https://images.shodan.io/?query=http

```
HTTP
```

To search for VNC you can search using **RFB** and for RTSP you simply search with **RTSP**.

The images can also be found using the main Shodan website or Shodan Maps by using the search query **has_screenshot:true**.

Exercises: Website

Exercise 1

Find the 4SICS website using Shodan.

> **Tip**: Check out Appendix B for a list of search filters.

Exercise 2

How many VNC services on the Internet allow anonymous access?

Exercise 3

How many IPs in Sweden are vulnerable to Heartbleed and still support SSLv3?

How many IPs are vulnerable to Heartbleed at your organization?

Exercise 4

Find all the industrial control systems in your town.

Exercise 5

Which RAT is most popular in the USA?

External Tools

Shodan Command-Line Interface

Getting Started

The **shodan** command-line interface is packaged with the official Python library for Shodan, which means if you're running the latest version of the library you already have access to the CLI. To install the new tool simply execute:

```
easy_install shodan
```

Once the tool is installed it has to be initialized with your API key:

```
shodan init YOUR_API_KEY
```

Visit https://account.shodan.io to retrieve the API key for your account.

alert

The **alert** command provides you the ability to **create**, **list**, **clear** and **remove** network alerts.

convert

Convert the compressed JSON file generated by Shodan into a different file format. At the moment it supports output to **kml** and **csv**.

count

Returns the number of results for a search query.

```
$ shodan count microsoft iis 6.0
5360594
```

download

Search Shodan and download the results into a file where each line is a JSON banner (see Appendix A).

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

The download command is what you should be using most often when getting results from Shodan since it lets you save the results and process them afterwards using the parse command. Because paging through results uses query credits, it makes sense to always store searches that you're doing so you won't need to use query credits for a search you already did in the past.

```
$ shodan download microsoft-data microsoft iis 6.0
Search query:                   microsoft iis 6.0
Total number of results:        5310596
Query credits left:             100000
Output file:                    microsoft-data.json.gz
  [#######-----------------------------]   20%  00:00:20
```

host

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

```
$ shodan host 189.201.128.250
```

```
189.201.128.250
Hostnames:                customer-250.xertix.com
City:                     Mexico
Country:                  Mexico
Organization:             Metro Net, S.A.P.I. de C.V.
Number of open ports:     1
Vulnerabilities:          Heartbleed

Ports:
    443 Fortinet FortiGate 50B or FortiWifi 80C firewall http config
        |-- SSL Versions: SSLv3, TLSv1, TLSv1.1, TLSv1.2
        |-- Diffie-Hellman Parameters:
                Bits:          1024
                Generator:     2
                Fingerprint:   RFC2409/Oakley Group 2
```

honeyscore

Check whether an IP address is a honeypot pretending to be an industrial control system.

```
$ shodan honeyscore 41.231.95.212
```

info

Obtain general information about your API plan, including how many query and scan credits you have remaining this month.

```
$ shodan info
Query credits available: 5102
Scan credits available: 249
```

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.

The following command outputs the IP address, port and organization in CSV format for the previously downloaded Microsoft-IIS data:

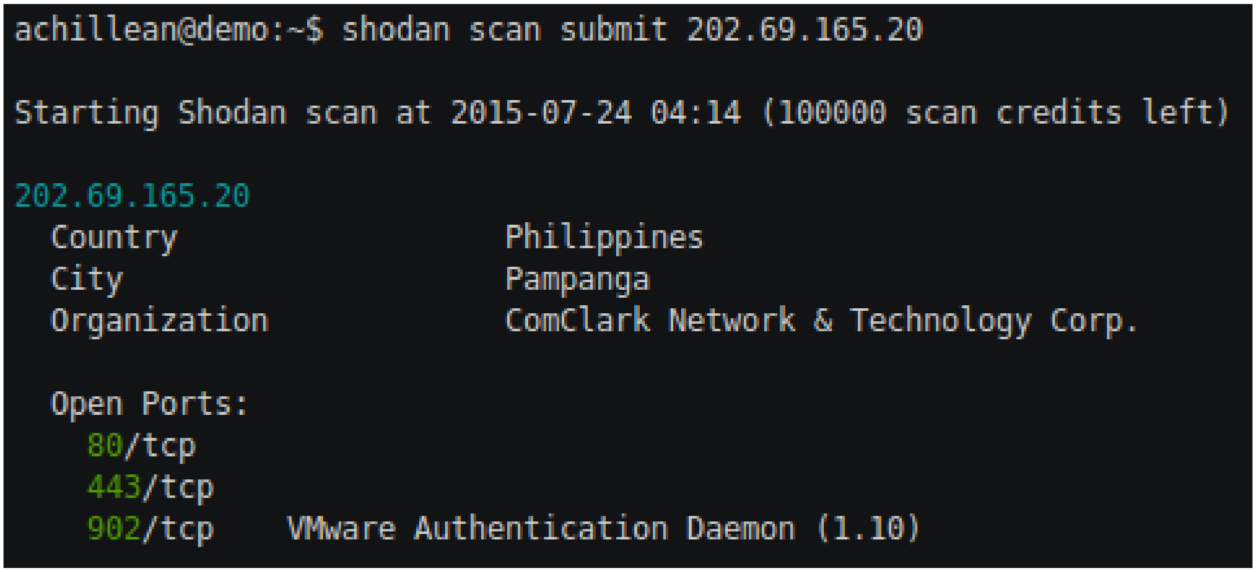
```
$ shodan parse --fields ip_str,port,org --separator , microsoft-data.json.gz
```

```
216.28.245.171,80,Web Force Systems,
103.41.16.147,80,
218.244.142.211,80,China Network Information Center,
81.22.98.166,80,Kriter Internet Hiz.Ltd.Sti.,
75.149.30.138,443,Comcast Business Communications,
23.108.235.233,80,Nobis Technology Group, LLC,
207.57.69.157,8080,Verio Web Hosting,
66.129.113.13,80,Peak 10,
168.143.6.120,8080,Verio Web Hosting,
218.0.3.56,80,China Telecom Ningbo,
104.202.81.231,80,
98.191.178.20,443,Cox Communications,
108.186.164.90,80,Peg Tech,
23.105.63.236,80,Nobis Technology Group, LLC,
67.227.184.237,8443,Smash Data Design,
107.163.173.34,80,
185.22.198.84,80,Nexto SAS,
72.29.22.40,80,Cybercon,
216.119.84.188,80,CrystalTech Web Hosting,
104.221.145.60,80,
198.171.51.81,8080,Verio Web Hosting,
209.10.173.10,443,Quality Technology Services, N.J., LLC,
```

scan

The `scan` command provides a few sub-commands but the most important one is `submit` which lets you perform network scans using Shodan.

```
$ shodan scan submit 202.69.165.20
```



```
achillean@demo:~$ shodan scan submit 202.69.165.20

Starting Shodan scan at 2015-07-24 04:14 (100000 scan credits left)

202.69.165.20
  Country                  Philippines
  City                     Pampanga
  Organization             ComClark Network & Technology Corp.

  Open Ports:
    80/tcp
    443/tcp
    902/tcp    VMware Authentication Daemon (1.10)
```

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.

For example, to search Microsoft IIS 6.0 and print out their IP, port, organization and hostnames use the following command:

```
$ shodan search --fields ip_str,port,org,hostnames microsoft iis 6.0
```

```
81.171.175.68   80      Star Technology Services Limited
178.73.238.43   80      Portlane Networks AB
113.245.76.199  5900    China Telecom HUNAN
149.210.160.163 80      Transip B.V.    nowarkrengelink.com
23.92.216.117   80      Res.pl Isp S.c. mailingrolout.com
202.69.233.212  443     Verio Web Hosting       kubota-rvc23-0727001.com
190.78.179.228  8080    CANTV Servicios, Venezuela      190-78-179-228.dyn.dsl.cantv.net
192.3.4.108     443     ColoCrossing    sxi.pw
160.246.182.223 80      Hayashi Telempu Co., Ltd.
198.104.15.120  443     Verio Web Hosting       wholesalechildrensclothing.com.au
208.64.139.67   80      Desync Networks 119-a.webmasters.com
212.227.51.115  443     1&1 Internet AG s535322526.online.de
75.98.17.22     443     Internap Network Services Corporation
178.208.77.241  81      McHost.Ru       v112059.vps.mcdir.ru
63.249.80.153   443     Cruzio  www12153.cruzio.com
87.243.209.223  8080    HotChilli Internet      static-87-243-209-223.adsl.hotchilli.net
183.89.74.87    81      3BB Broadband   mx-11-183.89.74-87.dynamic.3bb.co.th
178.236.77.90   80      Excellent Hosting Sweden AB
54.201.193.170  80      Amazon.com      ec2-54-201-193-170.us-west-2.compute.amazonaws.com
106.186.28.222  80      Linode, LLC     li608-222.members.linode.com
54.85.166.63    80      Merck and Co.   ec2-54-85-166-63.compute-1.amazonaws.com
208.131.128.136 80      WestHost        greenstreetstudios.org
```

stats

The `stats` command lets you print the facets for a search query.

For example, the following command shows the most popular countries where Apache web servers are located in:

```
$ shodan stats --facets country apache
Top 10 Results for Facet: country
US                              8,336,729
DE                              4,512,172
CN                              1,470,434
JP                              1,093,699
GB                                832,221
NL                                684,432
FR                                667,871
CA                                501,630
RU                                324,698
BR                                266,788
```

stream

The `stream` command provides access to the real-time stream of data that the Shodan crawlers collect.

```
achillean@demo:~$ shodan stream --help
Usage: shodan stream [OPTIONS]

  Stream data in real-time.

Options:
  --color / --no-color
  --fields TEXT          List of properties to output.
  --separator TEXT       The separator between the properties of the search
                         results.
  --limit INTEGER        The number of results you want to download. -1 to
                         download all the data possible.
  --datadir TEXT         Save the stream data into the specified directory as
                         .json.gz files.
  --ports TEXT           A comma-separated list of ports to grab data on.
  --quiet                Disable the printing of information to the screen.
  --streamer TEXT        Specify a custom Shodan stream server to use for
                         grabbing data.
  -h, --help             Show this message and exit.
```

The command supports many different flags, however there are 2 that are important to mention:

–datadir

The **–datadir** flag lets you specify a directory in which the streamed data should be stored. The files generated in the **–datadir** directory have the following naming convention:

```
YYYY-MM-DD.json.gz
```

A sample file name would be "2016-01-15.json.gz". Each day a new file is automatically generated as long as you keep the stream running. For example, the following command downloads all the data from the real-time stream and saves it in a directory called **/var/lib/shodan/**:

```
shodan stream --datadir /var/lib/shodan/
```

–limit

The **–limit** flag specifies how many results that should be downloaded. By default, the `stream` command runs forever until you exit the tool. However, if you're only interested in collecting a sample of data then the **–limit** flag ensures you gather a small amount of records. For example:

```
shodan stream --limit 100
```

The above command would connect to the Shodan real-time stream, print out the first 100 records that are received and then exit.

–ports

The **–ports** flag accepts a comma-separated list of ports to let you stream only records gathered from those ports. The following command prints out a stream of banners that were collected from services running on port 80 or 8080:

```
shodan stream --ports 80,8080
```

Example: Telnet Research

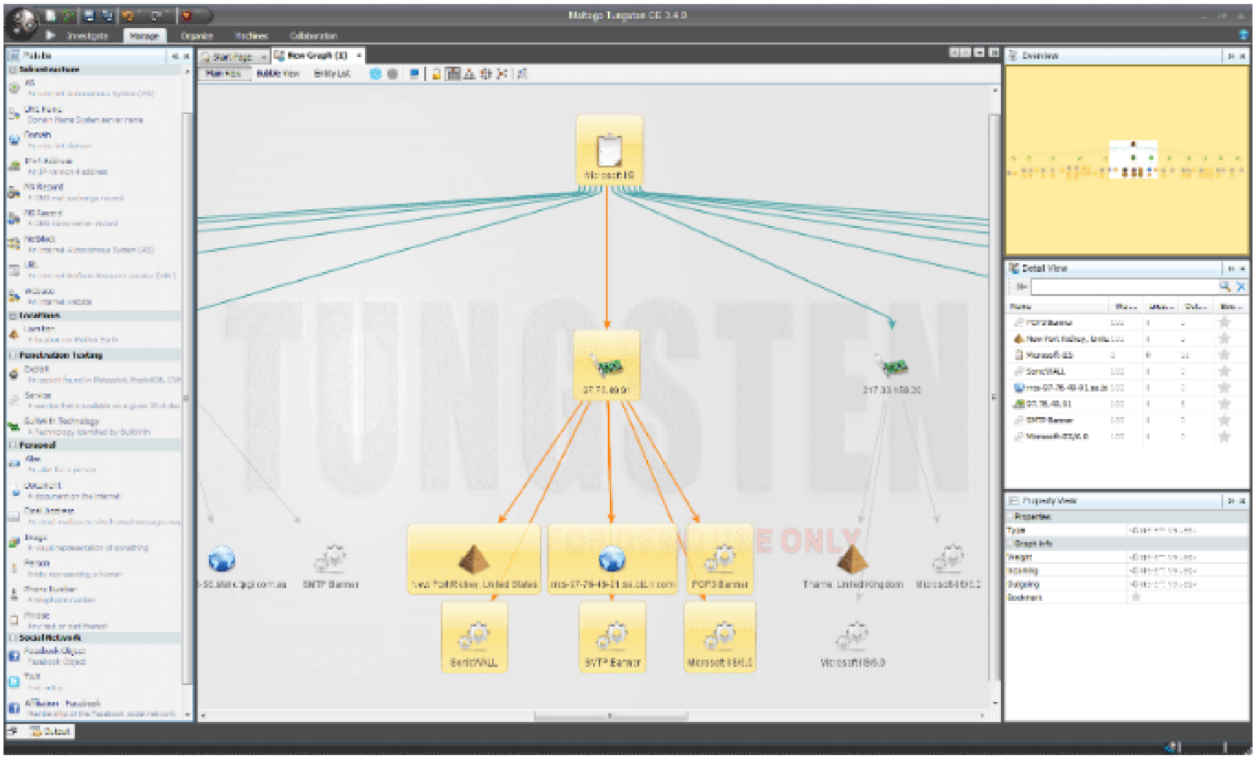
Lets assume we want to perform research into devices on the Internet running Telnet. As a starting point we can combine all of the aforementioned commands into the following:

```
mkdir telnet-data
shodan stream --ports 23,1023,2323 --datadir telnet-data/ --limit 10000
```

First, we create a directory called **telnet-data** to store the Telnet data. Then we request 10,000 records (**–limit 10000**) from the stream on common Telnet ports (**–ports 23,1023,2323**) and store the results in the previously created directory (**–datadir telnet-data/**).

Maltego Add-On

Maltego is an open source intelligence and forensics application; it lets you visually explore and correlate data from a variety of sources.



The Shodan add-on for Maltego provides 2 new entities (`Service` and `Exploit`) and 5 transforms:

- searchShodan
- searchShodanByDomain
- searchShodanByNetblock
- toShodanHost
- searchExploits

Browser Plug-Ins

There are plugins available for both Chrome[5] and Firefox[6] that let you see what services a website exposes.

[5] https://chrome.google.com/webstore/detail/shodan/jjalcfnidlmpjhdfepjhjbhnhkbgleap
[6] https://addons.mozilla.org/en-us/firefox/addon/shodan-firefox-addon/

Exercises: Command-Line Interface

Exercise 1

Download the IPs vulnerable to Heartbleed in Sweden and Norway using the Shodan CLI.

Filter out the results for Sweden and store them in a separate file.

> Note: Uncompress the file and look at the raw data to see the raw response from the Heartbleed test.

Exercise 2

Download 1,000 recent banners using the real-time stream and then map them using Google Maps.

> **Tip**: shodan convert

Exercise 3

Write a script to download a list of known malware IPs and block any outgoing traffic to them.

> **Tip**: iptables -A OUTPUT -d x.x.x.x -j DROP

Developer API

Shodan provides a developer API (https://developer.shdan.io/api) for programmatic access to the information that is collected. All of the websites and tools, including the main Shodan website, are powered by the API. Everything that can be done via the website can be accomplished from your own code.

The API is divided into 2 parts: REST API and Streaming API. The REST API provides methods to search Shodan, look up hosts, get summary information on queries and a variety of utility methods to make developing easier. The Streaming API provides a raw, real-time feed of the data that Shodan is currently collecting. There are several feeds that can be subscribed to, but the data can't be searched or otherwise interacted with; it's a live feed of data meant for large-scale consumption of Shodan's information.

> **Note**: Only users with an API subscription are able to access the Streaming API.

Usage Limits

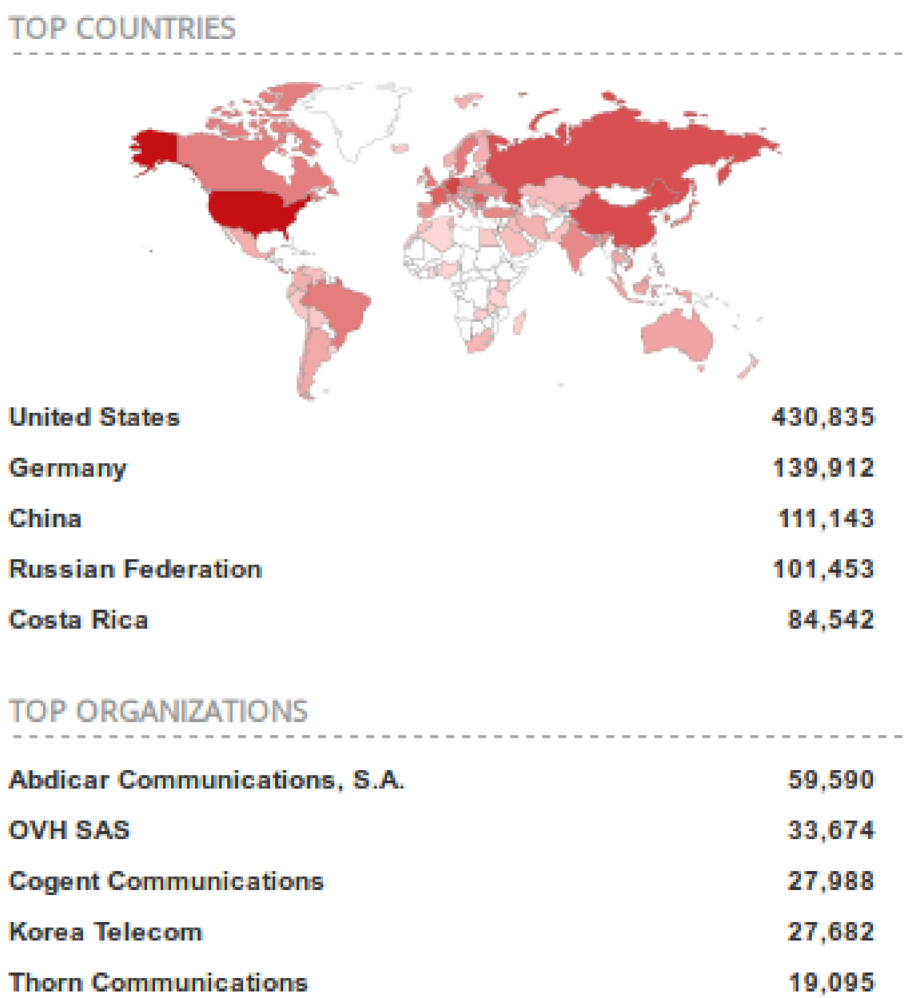
There are 3 methods of the API that get limited depending on your API plan:

1. **Searching** To limit the number of searches that can be performed per month Shodan uses **query credits**. 1 query credits is used when you **perform a search containing filters** or **go past the 1st page**. For example, if you search for "apache" that doesn't use any query credits. If you search for "apache country:US" that would use 1 query credit. Likewise, if you searched for the 2nd page of results for "apache" that would use 1 query credit. Finally, a search query for the 2nd page of "apache country:US" would also use up 1 query credit.
2. **Scanning** The on-demand scanning API uses **scan credits** to limit the number of hosts that you can request Shodan to scan every month. For every host that you request a scan of Shodan deducts 1 scan credit.
3. **Network Alerts** The number of IPs that can be monitored using alerts is limited based on your API subscription. Only paid customers have access to this feature. And you can't create more than 100 alerts on your account.

> **Important**: Query and scan credits get reset at the start of every month.

Introducing Facets

Facets provide aggregate information about a specific field of the banner you're interested in. Filters let you narrow down search results while facets let you get a *big picture* view of the results. For example, the main Shodan website uses facets to provide the statistics information on the left side of the search results:



A long list of facets are available (see **Appendix C**) and using the API you are in control of which facets you care about. For example, searching for `port:22` and faceting on the `ssh.fingerprint` facet will give you a breakdown of which SSH fingerprints are most commonly seen on the Internet. Facets are often the starting point for research into Internet-wide issues such as duplicate SSH keys, negligent hosting providers or country-wide security holes.

At the moment, facets are only available from the API and the Shodan command-line interface.

Getting Started

All the examples will be provided in Python and assume you have access to the command-line, though there are Shodan libraries/ clients available in other languages[7] as well.

To install the Shodan library for Python run the following command:

```
easy_install shodan
```

If you already have it installed and want to upgrade to the latest version:

```
easy_install -U shodan
```

Initialization

The first thing that always has to be done is initializing the Shodan API object:

```
import shodan
api = shodan.Shodan('YOUR API KEY')
```

Where **YOUR API KEY** is the API key for you account which you can obtain from:

https://account.shodan.io

Search

Now that we have our API object all good to go, we're ready to perform a search:

```
# Wrap the request in a try/ except block to catch errors
try:
        # Search Shodan
        results = api.search('apache')

        # Show the results
        print 'Results found: %s' % results['total']
        for result in results['matches']:
                print 'IP: %s' % result['ip_str']
                print result['data']
                print ''
except shodan.APIError, e:
        print 'Error: %s' % e
```

[7] https://developer.shodan.io/api/clients

Stepping through the code, we first call the `Shodan.search()` method on the `api` object which returns a dictionary of result information. We then print how many results were found in total, and finally loop through the returned matches and print their IP and banner. Each page of search results contains up to 100 results.

There's a lot more information that gets returned by the function. See below for a shortened example dictionary that `Shodan.search` returns:

```
{
	'total': 8669969,
	'matches': [
		{
			'data': 'HTTP/1.0 200 OK\r\nDate: Mon, 08 Nov 2010 05:09:59 GMT\r\nSer...
			'hostnames': ['pl4t1n.de'],
			'ip': 3579573318,
			'ip_str': '89.110.147.239',
			'os': 'FreeBSD 4.4',
			'port': 80,
			'timestamp': '2014-01-15T05:49:56.283713'
		},
		...
	]
}
```

See Appendix A for a complete list of properties that the banner may contain.

Important: By default, a few of the large fields in the banner such as "html" get truncated to reduce bandwidth usage. If you want to retrieve all the information simply disable minification using `minify=False`. For example, the following search query for anonymous VNC services would ensure all information is returned:

```
results = api.search('has_screenshot:true', minify=False)
```

It's also good practice to wrap all API requests in a try/ except clause, since any error will raise an exception. But for simplicity's sake, I will leave that part out from now on.

The above script only outputs the results from the 1st page of results. To get the 2nd page of results or more simply use the `page` parameter when doing the search request:

```
results = api.search('apache', page=2)
```

Or if you want to simply loop over all possible results there's a method to make your life easier called `search_cursor()`

```
for banner in api.search_cursor('apache'):
        print banner['ip_str'] # Print out the IP address for each banner
```

> **Important**: The `search_cursor()` method only returns the banners and doesn't let you use facets. Only use it to loop over results.

Host Lookup

To see what Shodan has available on a specific IP we can use the `Shodan.host()` function:

```
# Lookup the host
host = api.host('217.140.75.46')

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

# Print all banners
for item in host['data']:
        print """
                Port: %s
                Banner: %s

        """ % (item['port'], item['data'])
```

By default, Shodan only returns information on the host that was recently collected. If you would like to get a full history of an IP address, include the `history` parameter. For example:

```
host = api.host('217.140.75.46', history=True)
```

The above would return all banners, including for services that may no longer be active on the host.

Scanning

Shodan crawls the Internet at least once a month, but if you want to request Shodan to scan a network immediately you can do so using the on-demand scanning capabilities of the API.

Unlike scanning via a tool such as Nmap, the scanning with Shodan is done asynchronously. This means that after you submit a request to Shodan you don't get back the results immediately. It is up to the developer to decide how the results of the scan should be gathered: by looking up the IP information, searching Shodan or subscribing to the real-time stream. The Shodan command-line interface creates a temporary network alert after a scan was initiated and then waits for results to come through the real-time stream.

```
scan = api.scan('198.20.69.0/24')
```

It's also possible to submit a list of networks at once by providing a list of addresses in CIDR notation:

```
scan = api.scan(['198.20.49.30', '198.20.74.0/24'])
```

After submitting a scan request the API will return the following information:

```
{
	'id': 'R2XRT5HH6X67PFAB',
	'count': 1,
	'credits_left': 5119
}
```

The object provides a unique `id` that you can use for tracking purposes, the total `count` of IPs that were submitted for scanning and finally how many scan credits are left (`credits_left`).

Real-Time Stream

The Streaming API is an HTTP-based service that returns a real-time stream of data collected by Shodan. It doesn't provide any search or lookup capabilities, it is simply a feed of everything that is gathered by the crawlers.

For example, here is a script that outputs a stream of banners from devices that are vulnerable to FREAK (CVE-2015-0204):

```
def has_vuln(banner, vuln):
    if 'vulns' in banner['opts'] and vuln in banner['opts']['vulns']:
        return True
    return False

for banner in api.stream.banners():
    if has_vuln(banner, 'CVE-2015-0204'):
        print banner
```

To save space and bandwidth many properties in the banner are optional. To make working with optional properties easier it is best to wrap access to properties in a function. In the above example, the `has_vuln()` method checks whether the service is vulnerable for the provided CVE.

> **Note**: Regular API subscriptions only have access to 1% of the feed. 100% access is available to data license customers only.

Network Alert

A network alert is a real-time feed of data that is being collected by Shodan for a network range. To get started with network alerts requires 2 steps:

Creating a Network Alert

To create a network alert you need to provide a `name` and a `network range`. The name should be descriptive to let you know what the alert is monitoring or why it was created.

```
alert = api.create_alert('Production network', '198.20.69.0/24')
```

As with the `scan()` method you can also provide a list of network ranges to monitor:

```
alert = api.create_alert('Production and Staging network', [
        '198.20.69.0/24',
        '198.20.70.0/24',
])
```

> **Note**: Only a limited number of IPs can be monitored using network alerts and an account can't have more than 100 alerts active.

A useful trick when combining network alerts with the scanning API is to set an expiration for the alert:

```
alert = api.create_alert('Temporary alert', '198.20.69.0/24', expires=60)
```

The above alert would be active for `60` seconds and then expire, at which point the alert can't be used any more.

Upon successfully creating an alert, the API will return the following object:

```
{
        "name": "Production network",
        "created": "2015-10-17T08:13:58.924581",
        "expires": 0,
        "expiration": null,
        "filters": {
                "ip": ["198.20.69.0/24"]
        },
        "id": "EPGWQG5GEELV4799",
        "size": 256
}
```

Subscribing

Once an alert has been created it is ready to be used as a real-time stream of data for that network.

```
for banner in api.stream.alert(alert['id']):
        print banner
```

As with the regular, real-time stream the `alert()` method provides an iterator where each item is a banner as it's being collected by the Shodan crawlers. The only argument that the `alert()` method requires is the alert ID that was returned when creating the network alert.

Using the Shodan Command-Line Interface

We will quickly examine how the above Python code could be implemented using the Shodan CLI[8] instead. Let's start off by clearing out any existing alerts:

Warning

The **clear** command removes all alerts that have been created on your account.

[8] https://cli.shodan.io

```
$ shodan alert clear
Removing Scan: 198.20.69.0/24 (ZFPSZCYUKVZLUT4F)
Alerts deleted
```

And confirm that there aren't any existing alerts anymore:

```
$ shodan alert list
You haven't created any alerts yet.
```

Now it's time to create a new network alert:

```
$ shodan alert create "Temporary alert" 198.20.69.0/24
Successfully created network alert!
Alert ID: ODMD34NFPLJBRSTC
```

The final step is to subscribe to the network alert and store the data it returns. To stream results for the created alert we give the alert ID **ODMD34NFPLJBRSTC** to the **stream** command:

```
$ mkdir alert-data
$ shodan stream --alert=ODMD34NFPLJBRSTC --datadir=alert-data
```

In the above command we are streaming results for the **–alert** with the ID of **ODMD34NFPLJBRSTC** and the results will be stored in a directory called **alert-data**. Every day a new file will be generated in the **alert-data** directory which contains the banners gathered for the day. I.e. we don't need to take care of rotating files, the **stream** command will take care of that for us so after a few days the directory will look like:

```
$ ls alert-data
2016-06-05.json.gz
2016-06-06.json.gz
2016-06-07.json.gz
```

Example: Public MongoDB Data

MongoDB[9] is a popular NoSQL[10] database and for a long time it didn't come with any authentication. This has resulted in many instances of MongoDB being publicly accessible on the Internet. Shodan grabs a banner for these databases that contains a lot of information about the data stored. Following is an excerpt from the banner:


```
MongoDB Server Information
...
{
    "ok": 1.0,
    "tokumxAuditVersion": "unknown",
    "bits": 64,
    "tokukvVersion": "unknown",
    "tokumxVersion": "2.0.2",
    "javascriptEngine": "V8",
    "version": "2.4.10",
    "versionArray": [
        2,
        4,
        10,
        0
    ],
    "debug": false,
    "compilerFlags": "-fPIC -fno-strict-aliasing -ggdb -Wall -Wsign-compare -Wno\
-unknown-pragmas -Winvalid-pch -pipe -Wnon-virtual-dtor -Woverloaded-virtual -Wn\
o-unused-local-typedefs -fno-builtin-memcmp -O3",
    "maxBsonObjectSize": 16777216,
    "sysInfo": "Linux vps-vivid-x64-04 2.6.32-042stab106.6 #1 SMP Mon Apr 20 14:\
48:47 MSK 2015 x86_64 x86_64 x86_64 GNU/Linux BOOST_LIB_VERSION=1_55",
    "loaderFlags": "       ",
    "gitVersion": "unknown"
},
...
```


Basically, the banner is made up of a header that says "MongoDB Server Information" followed by 3 JSON objects that are separated by commas. Each JSON object contains different information about the database and I recommend you check out a full banner on Shodan (it's very long) by searching for:

[9] https://www.mongodb.com
[10] https://en.wikipedia.org/wiki/NoSQL

```
product:MongoDB
```

Let's use the banner information to determine which database names are most popular and how much data is publicly exposed on the Internet! The basic workflow will be to:

1. Download all MongoDB banners
2. Process the downloaded file and output a list of top 10 database names as well as the total data size

Downloading the data is simple using the Shodan command-line interface[11]:

```
shodan download --limit -1 mongodb-servers.json.gz product:mongodb
```

The above command says to download all results (**–limit -1**) into a file called **mongodb-servers.json.gz** for the search query **product:mongodb**. You can also download the results of the command from the **Extras** section of the book on Leanpub. Now we just need a simple Python script to process the Shodan data file. To easily iterate over the file we're going to use the **shodan.helpers.iterate_files()** method:

```
import shodan.helpers as helpers
import sys

# The datafile is the 1st argument to the command
datafile = sys.argv[1]

for banner in helpers.iterate_files(datafile):
    # Now we have the banner
```

Since each banner is just JSON with some added header, lets process the banner into a native Python dictionary using the **simplejson** library:

```
# Strip out the MongoDB header added by Shodan
data = banner['data'].replace('MongoDB Server Information\n', '').split('\n},\n'\
)[2]

# Load the database information
data = simplejson.loads(data + '}')
```

The only thing that's left is keeping track of the total amount of data that's exposed and the most popular database names:

[11] https://cli.shodan.io

```
total_data = 0
databases = collections.defaultdict(int)

...

    # Then in the loop
    # Keep track of how much data is publicly accessible
    total_data += data['totalSize']

    # Keep track of which database names are most common
    for db in data['databases']:
        databases[db['name']] += 1
```

Python has a useful **collections.defaultdict** class that automatically creates a default value for a dictionary key if the key doesn't yet exist. And we just access the **totalSize** and **databases** property of the MongoDB banner to gather the information we care about. Finally, we just need to output the actual results:

```
print('Total: {}'.format(humanize_bytes(total_data)))

counter = 1
for name, count in sorted(databases.iteritems(), key=operator.itemgetter(1), rev\
erse=True)[:10]:
    print('#{}\t{}: {}'.format(counter, name, count))
    counter += 1
```

First, we print the total amount of data that's exposed and we're using a simple **humanize_bytes()** method to convert bytes into human-readable format of GB/ MB/ etc. Second, we loop sort the **databases** collection in **reverse** order by the number of times that a certain database name was seen (**key=operator.itemgetter(1)**) and get the top 10 results (**[:10]**).

Below is the full script that reads a Shodan data file and analyzes the banner:

```
import collections
import operator
import shodan.helpers as helpers
import sys
import simplejson

def humanize_bytes(bytes, precision=1):
    """Return a humanized string representation of a number of bytes.
```

```
    Assumes `from __future__ import division`.

    >>> humanize_bytes(1)
    '1 byte'
    >>> humanize_bytes(1024)
    '1.0 kB'
    >>> humanize_bytes(1024*123)
    '123.0 kB'
    >>> humanize_bytes(1024*12342)
    '12.1 MB'
    >>> humanize_bytes(1024*12342,2)
    '12.05 MB'
    >>> humanize_bytes(1024*1234,2)
    '1.21 MB'
    >>> humanize_bytes(1024*1234*1111,2)
    '1.31 GB'
    >>> humanize_bytes(1024*1234*1111,1)
    '1.3 GB'
    """
    abbrevs = (
        (1<<50L, 'PB'),
        (1<<40L, 'TB'),
        (1<<30L, 'GB'),
        (1<<20L, 'MB'),
        (1<<10L, 'kB'),
        (1, 'bytes')
    )
    if bytes == 1:
        return '1 byte'
    for factor, suffix in abbrevs:
        if bytes >= factor:
            break
    return '%.*f %s' % (precision, bytes / factor, suffix)

total_data = 0
databases = collections.defaultdict(int)
for banner in helpers.iterate_files(sys.argv[1]):
    try:
        # Strip out the MongoDB header added by Shodan
        data = banner['data'].replace('MongoDB Server Information\n', '').split(\
'\n},\n')[2]
```

```
        # Load the database information
        data = simplejson.loads(data + '}')

        # Keep track of how much data is publicly accessible
        total_data += data['totalSize']

        # Keep track of which database names are most common
        for db in data['databases']:
            databases[db['name']] += 1
    except Exception, e:
        pass

print('Total: {}'.format(humanize_bytes(total_data)))

counter = 1
for name, count in sorted(databases.iteritems(), key=operator.itemgetter(1), rev\
erse=True)[:10]:
    print('#{}\t{}: {}'.format(counter, name, count))
    counter += 1
```

Here's a sample output of the script:

```
Total: 1.8 PB
#1      local: 85845
#2      admin: 67648
#3      test: 24983
#4      s: 5121
#5      config: 4329
#6      proxy: 2045
#7      research: 2007
#8      seolib_new: 2001
#9      traditional: 1998
#10     simplified: 1998
```

Exercises: Shodan API

Exercise 1

Write a script to monitor a network using Shodan and send out notifications.

Exercise 2

Write a script to output the latest images into a directory.

> **Tip**: Images are encoded using base64. Python can easily decode it into binary using: image_string.decode('base64')

Industrial Control Systems

In a nutshell, industrial control systems (ICS) are computers that control the world around you. They're responsible for managing the air conditioning in your office, the turbines at a power plant, the lighting at the theatre or the robots at a factory.

Research conducted from 2012 through 2014 by Project SHINE[12] (SHodan INtelligence Extraction) indicates there are at least 2 million publicly accessible devices related to ICS on the Internet. The first dataset containing 500,000 ICS devices was sent in 2012 to the ICS-CERT. The ICS-CERT determined that roughly 7,200 out of the 500,000 were critical infrastructure[13] in the United States. And with the demand for increased connectivity in everything that number is expected to rise. There have been efforts to secure these devices by taking them offline or patching flaws, but it's a challenging problem and there isn't an easy solution.

Common Abbreviations

Before getting into the protocols and how to find ICS devices, here are a few common abbreviations that are useful to know:

BMS	Building Management System
DCS	Distributed Control System
HMI	Human Machine Interface
ICS	Industrial Control System
PLC	Programmable Logic Controller
RTU	Remote Terminal Unit
SCADA	Supervisory Control and Data Acquisition (a subset of ICS)
VNC	Virtual Network Computing

Protocols

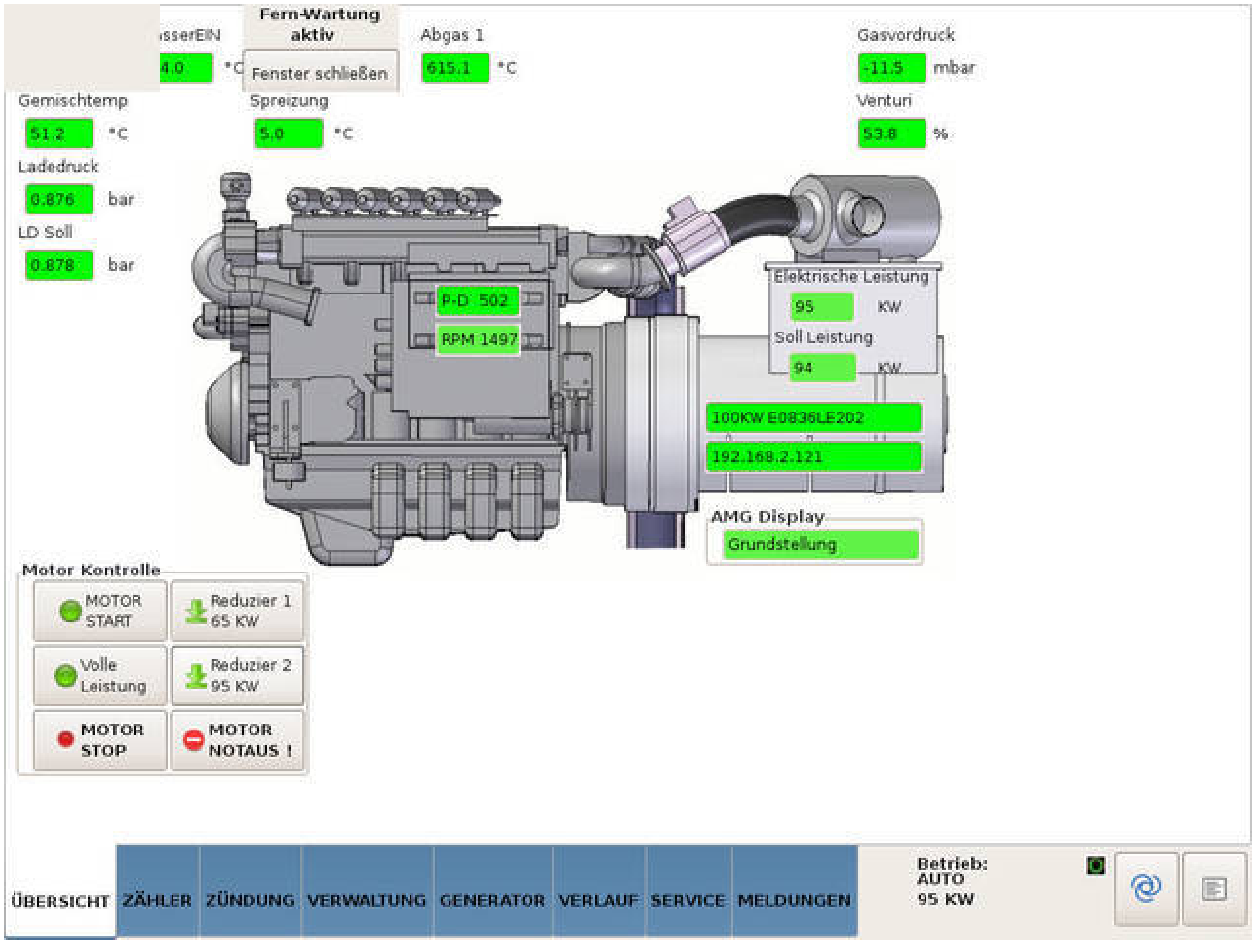
There are 2 different ways of identifying control systems on the Internet:

Non-ICS protocols used in an ICS environment

The majority of the ICS findings on Shodan are discovered by searching for web servers or other popular protocols that aren't directly linked to ICS but may be seen on an ICS network. For example:

[12] http://www.slideshare.net/BobRadvanovsky/project-shine-findings-report-dated-1oct2014
[13] https://ics-cert.us-cert.gov/sites/default/files/Monitors/ICS-CERT_Monitor_Oct-Dec2012.pdf

a web server running on an HMI or a Windows computer running unauthenticated remote desktop while connected to an ICS. These protocols provide you with a visual view of the ICS but they usually have some form of authentication enabled.



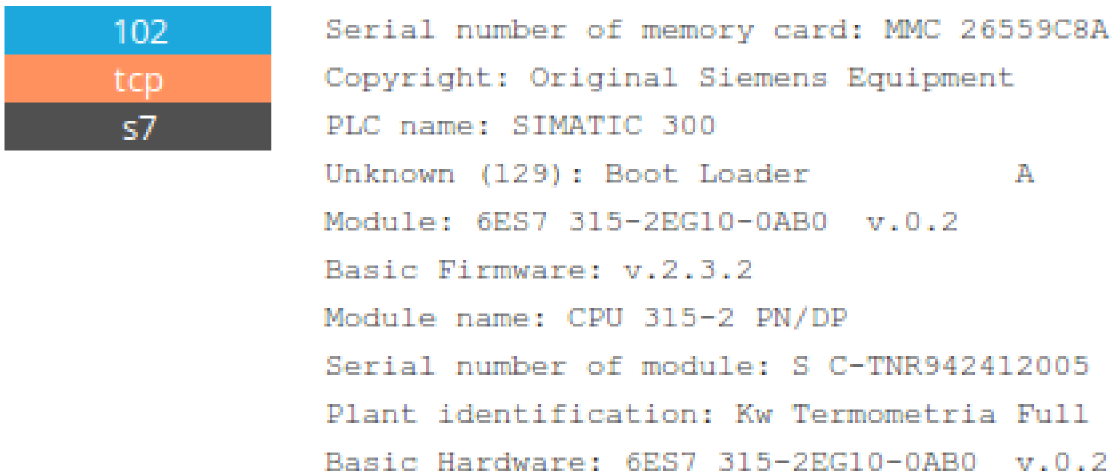
The above is an HMI for an engine exposed via an unauthenticated VNC connection found on Shodan Images[14].

ICS protocols

These are the raw protocols that are used by the control systems. Every ICS protocol has its own unique banner but there's one thing they all have in common: they don't require any authentication. This means that if you have remote access to an industrial device you automatically have the ability to arbitrarily read and write to it. However, the raw ICS protocols tend to be proprietary and hard to develop with. This means that it's easy to check whether a device supports an ICS protocol using Shodan but hard to actually interact with the control system.

[14] https://images.shodan.io

The following banner describes a Siemens S7 PLC, note that it contains a lot of detailed information about the device including its serial number and location:



Securing Internet-Connected ICS

The majority of ICS banners don't contain information on where the device is located or who owns the control system. This makes it exceedingly difficult to secure the device and is one of the main reasons that they continue to stay online after years of research into their online exposure.

If you discover a control system that looks critical, belongs to a government or otherwise shouldn't be online please notify the ICS-CERT[15]

Use Cases

Assessing ICS for the USA

You've been tasked with generating a quick presentation on the exposure of industrial control systems for the USA. To get started, lets first get a general idea of what's out there using the main Shodan website:

https://www.shodan.io/search?query=category%3Aics[16]

This returns a list of all devices running ICS protocols on the Internet. However, there are a lot of webservers and other protocols (SSH, FTP etc.) running on the same ports as industrial control systems which we need to filter out:

[15] https://ics-cert.us-cert.gov/Report-Incident?

[16] https://www.shodan.io/search?query=category%3Aics

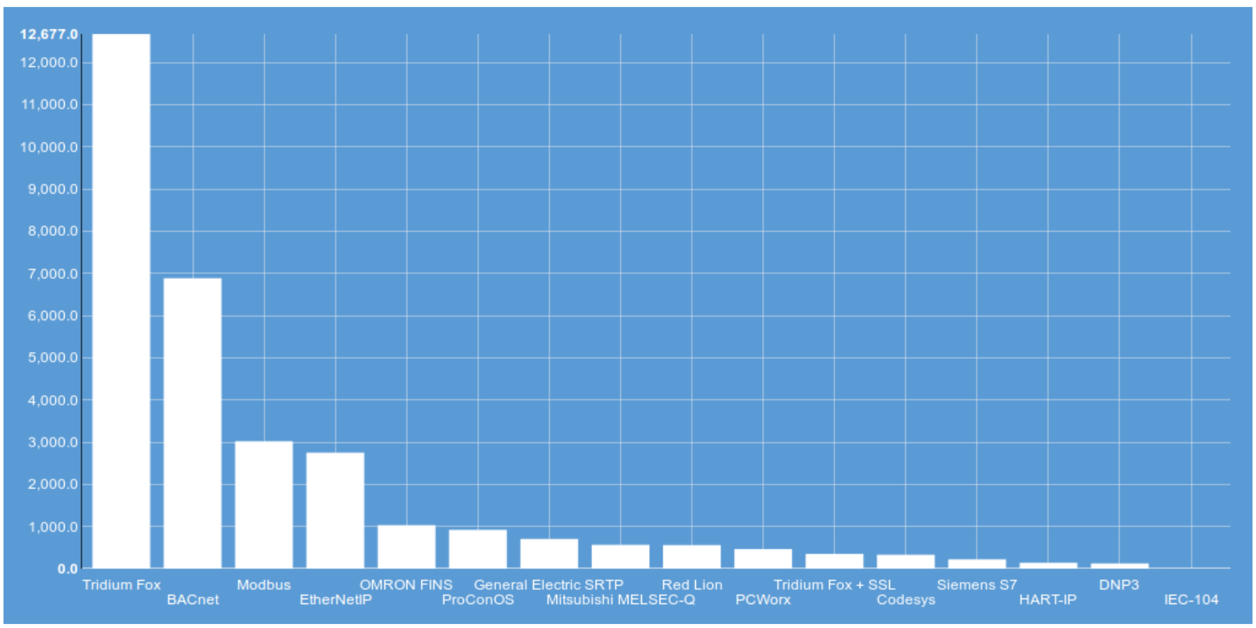
https://www.shodan.io/search?query=category%3Aics+-http+-html+-ssh+-ident[17]

> **Note**: if you have enterprise-level access you can use **tag:ics** instead of the above search query.

Now we have a filtered list of devices running insecure ICS protocols. Since the focus of the presentation will be on the USA, it's time to narrow the results to only IPs in the USA:

https://www.shodan.io/search?query=category%3Aics+-http+-html+-ssh+-ident+country%3Aus[18]

To get a big picture view of the data and have some charts to work with we can generate a free report[19]. This provides us with a better understanding of which ICS protocols are seen on the Internet in the US:

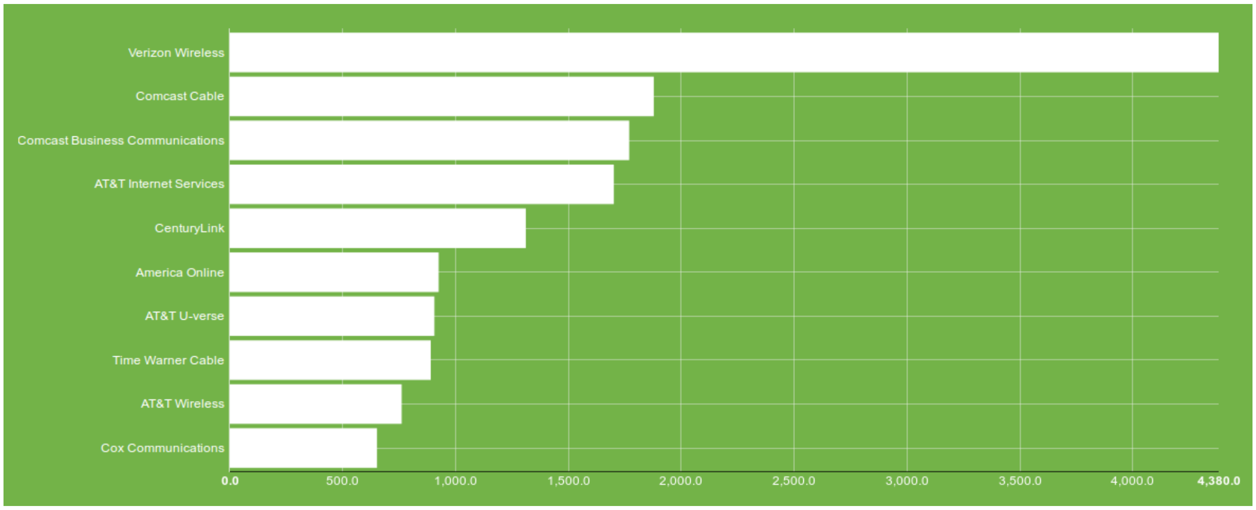


Tridium's Fox protocol, used by their Niagara framework, is the most popular ICS protocol in the US followed by BACnet and Modbus. The data shows that the majority of exposed devices are BMS used in offices, factories, stadiums, auditoriums and various facilities.

The above chart was saved as an image using Nimbus Screen Capture[20] on Firefox, but you can also use the Awesome Screenshot Minus[21] plug-in for Chrome.

The report[22] also highlights a common issue with ICS on the Internet: the majority of them are on mobile networks. This makes it especially difficult to track down and secure these devices.

[17]https://www.shodan.io/search?query=category%3Aics+-http+-html+-ssh+-ident
[18]https://www.shodan.io/search?query=category%3Aics+-http+-html+-ssh+-ident+country%3Aus
[19]https://www.shodan.io/report/mHpcVgdq
[20]https://addons.mozilla.org/en-US/firefox/addon/nimbus-screenshot/
[21]https://chrome.google.com/webstore/detail/awesome-screenshot-minus/bnophbnknjcjnbadhhkciahanapffepm?hl=en
[22]https://www.shodan.io/report/mHpcVgdq



At this point, the data shows us the following:

1. There are at least 65,000 ICS on the Internet exposing their raw, unauthenticated interfaces
2. Nearly half of them (~31,000) are in the US alone
3. Buildings are the most commonly seen type of ICS
4. Mobile networks host the largest amount

Further Reading

1. Distinguishing Internet-Facing Devices using PLC Programming Information[23]
2. NIST Special Publication - Guide to Industrial Control Systems Security[24]
3. Quantitatively Assessing and Visualizing Industrial System Attack Surfaces[25]

[23]https://www.hsdl.org/?abstract&did=757013
[24]http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-82r2.pdf
[25]https://www.cl.cam.ac.uk/~fms27/papers/2011-Leverett-industrial.pdf

Identifying Honeypots

Honeypots have become an increasingly popular and useful tool in understanding attackers. I've seen many misconfigured honeypots while scanning the Internet, here are a few tips to identify them or mistakes to avoid when setting them up.

What is a honeypot?

A honeypot is a device that pretends to be something it actually isn't for the purpose of logging and monitoring network activity. In the case of control systems, an ICS honeypot is a regular computer that pretends to be a control system such as a factory or power plant. They are used to collect information on attackers, including which networks the attackers are targeting, what tools they're using and many other useful insights that help defenders harden their network.

In recent years, honeypots have been used to measure the number of attacks that have been attempted against industrial control systems connected to the Internet. However, it is critically important to understand proper honeypot deployment before trying to gather the data. Many people misconfigure their honeypots and I will outline how those mistakes make it trivial to determine whether a device is a real control system or a honeypot.

The most popular and de-facto honeypot used to simulate industrial control systems is Conpot[26]. The software is well-written and extremely powerful when properly configured. Most of the examples and discussion will be using Conpot but the principles apply to all honeypot software.

Why Detect Them?

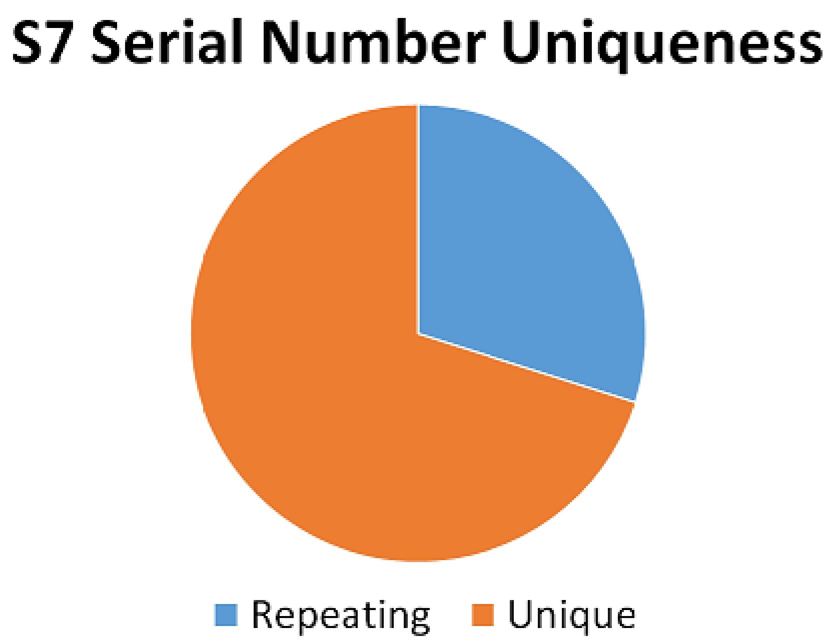
The data that honeypots generate is only as good as their deployment. If we want to make informed decisions about who is attacking control systems we have to ensure the data is being gathered from realistic honeypots. Sophisticated attackers won't be fooled by honeypots that are poorly configured. It's important to raise awareness for common pitfalls when deploying honeypots to improve the quality of data being collected.

Default Configurations

The most common mistake that people make when deploying honeypots is using the default configuration. All default configurations return the same banner, including identical serial numbers, PLC names and many other fields that you would expect to vary from IP to IP.

I first realized how common this problem is soon after doing the first Internet scan for Siemens S7:

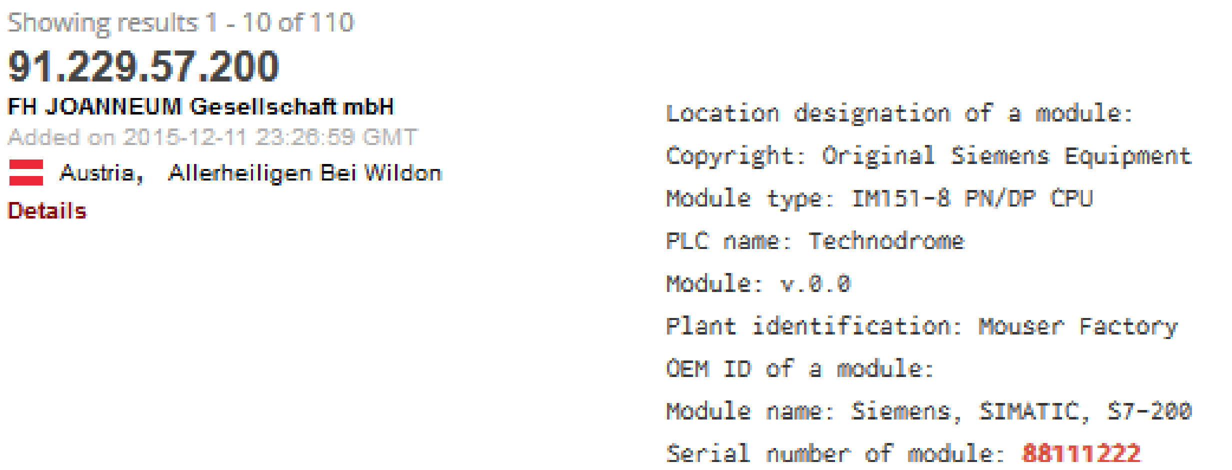
[26] https://github.com/mushorg/conpot/

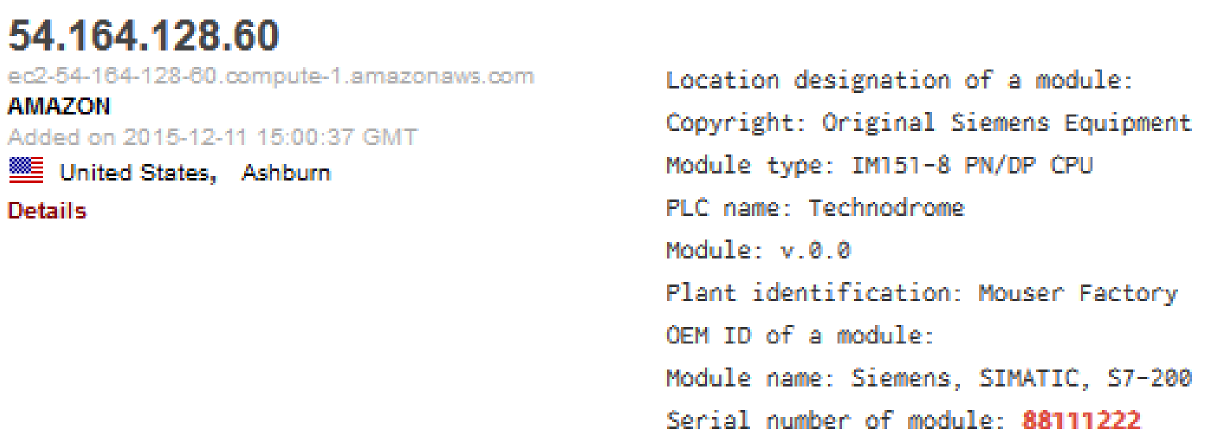


30% of the serial numbers in the results were present in more than one banner. It doesn't mean that all of the duplicate serial numbers are honeypots but it's a good starting point for investigation.

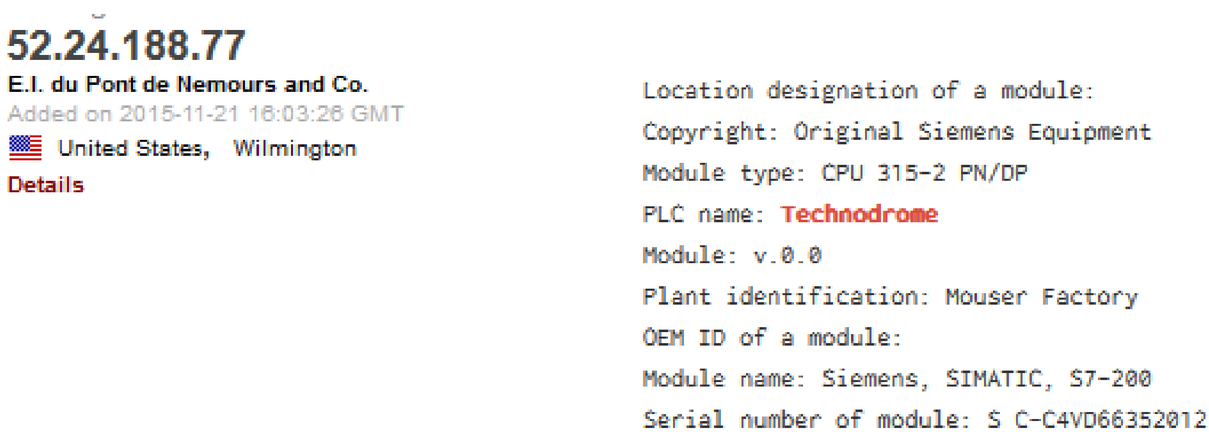
In the case of S7, the most popular serial number seen on the Internet is 88111222[27] which is the default serial number for Conpot.

[27]https://www.shodan.io/search?query=port%3A102+88111222





Searching by the serial number makes it trivial to locate instances of Conpot on the Internet. And make sure to also change the other properties of the banner, not just the serial number:



The above user changed the serial number to a unique value but failed to change the PLC name (**Technodrome**) and the plant identification (**Mouser Factory**). Every honeypot instance must have unique values in order to evade honeypot detection techniques.

History Matters

The honeypot has to be deployed properly from day 1 otherwise the banner history for the device will reveal it as a honeypot. For example:

```
Location designation of a module:
Copyright: Original Siemens Equipment
Module type: IM151-8 PN/DP CPU
PLC name: PG[random.randint(0,1) f
Module: v.0.0
Plant identification: Power Generation One
OEM ID of a module:
Module name: Siemens, SIMATIC, S7-200
Serial number of module: 8675309
```

The above is a banner pretending to be a Siemens S7 PLC. However, there was an error in the template generating the banner and instead of showing a valid PLC name it shows the template's **random.randint(0,1)** method. Shodan has indexed this banner and even if the bug is fixed in the future a user could look up the history for this IP and see that it used to have an invalid S7 banner.

A sample Shodan API request for the history of an IP:

```
host = api.host('xxx.xxx.xxx.xxx', history=True)
```

Emulate Devices, Not Services

Keep it simple, don't try to emulate too many services at once. A honeypot should emulate a device and most real devices don't run MongoDB, DNP3, MySQL, Siemens S7, Kamstrup, ModBus, Automated Tank Gauge, Telnet and SSH on the same IP.

Ports

22 80 102 161 502 623 3389 8649 20000 27017

Think about how the device is configured in the real-world and then emulate it, don't run every possible service simply because it's possible.

In code, you could use the number of ports as a metric:

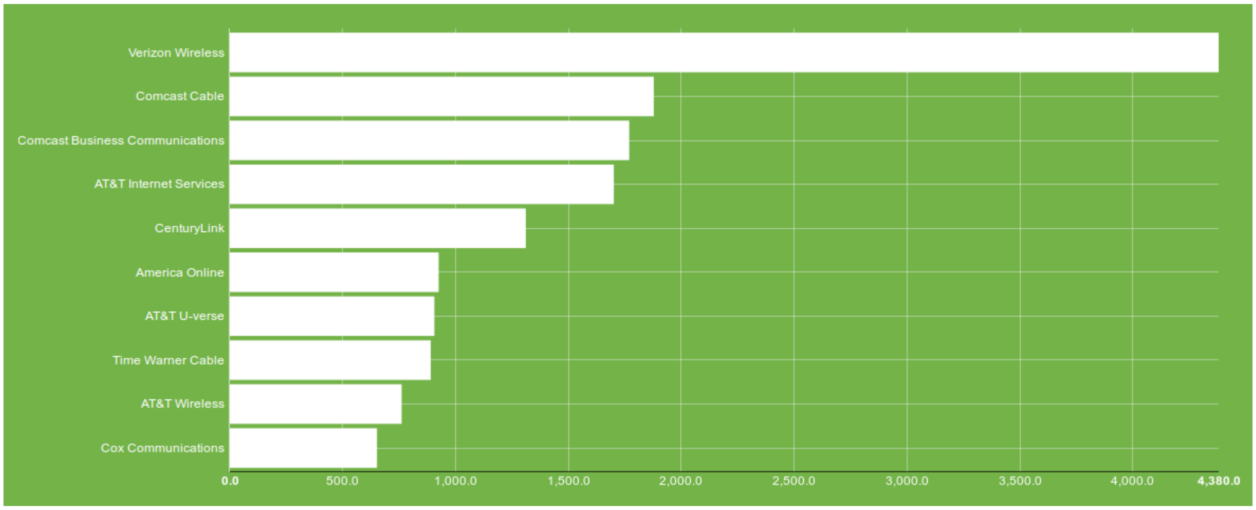
```
# Get information about the host
host = api.host('xxx.xxx.xxx.xxx')

# Check the number of open ports
if len(host['ports']) > 10:
        print('{} looks suspicious'.format(host['ip_str']))
else:
        print('{} has few ports open'.format(host['ip_str']))
```

Location, Location, Location

It isn't just the software that needs to be properly configured, a honeypot also has to be hosted on a network that could reasonably have a control system. Putting a honeypot that simulates a Siemens PLC in the Amazon cloud doesn't make any sense. Here are a few of the popular cloud hosting providers that should be avoided when deploying an ICS honeypot:

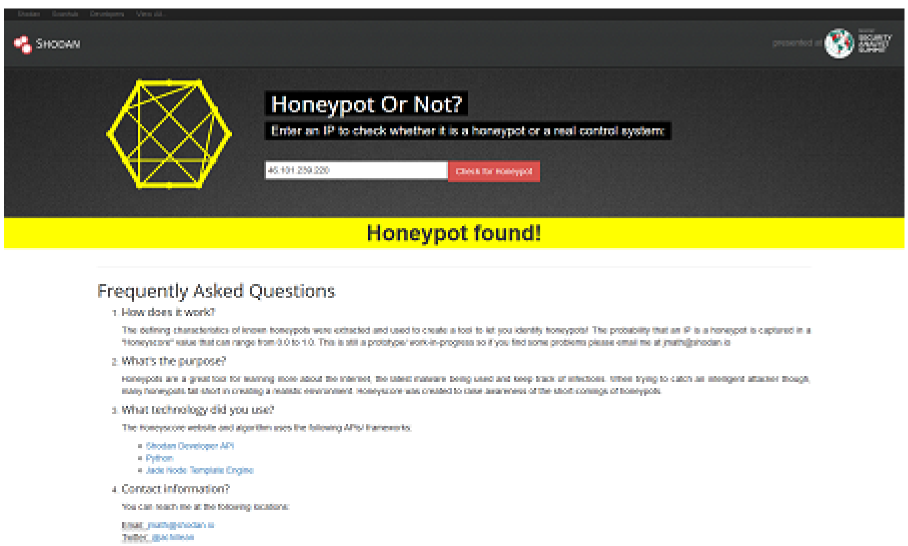
1. Amazon EC2
2. Rackspace
3. Digital Ocean
4. Vultr
5. Microsoft Azure
6. Google Cloud

For realistic deployment, look at the most popular ISPs in Shodan for publicly accessible ICS. In general, it is better to put the honeypot in the IP space of a residential ISP. The following organizations are the common locations in the USA:



Honeyscore

I developed a tool called Honeyscore[28] that uses all of the aforementioned methods as well as machine learning to calculate a **honeyscore** and determine whether an IP is a honeypot or not.



Simply enter the IP address of a device and the tool will perform a variety of checks to see whether it is a honeypot.

Tag: honeypot

The machine learning algorithm that powers Honeyscore has been ported to the crawlers so they're able to determine ICS honeypots as they collect data. In addition, we've also added detection for

[28]https://honeyscore.shodan.io

popular web application honeypots and a few Elastic honeypots. If a banner belongs to a known honeypot, then the **tags** property will contain the **honeypot** tag. The crawlers will also provide the name of the identified honeypot in the product property.

Further Reading

1. Wikipedia article on honeypots[29]
2. Breaking Honeypots for Fun and Profit (Video)[30]

[29]https://en.wikipedia.org/wiki/Honeypot_%28computing%29
[30]https://www.youtube.com/watch?v=Pjvr25lMKSY

Appendix A: Banner Specification

For the latest list of fields that the banner contains please visit the online documentation[31].

A banner may contain the following properties/ fields:

General Properties

Name	Description	Example
asn	Autonomous system number	AS4837
data	Main banner for the service	HTTP/1.1 200…
ip	IP address as an integer	493427495
ip_str	IP address as a string	199.30.15.20
ipv6	IPv6 address as a string	2001:4860:4860::8888
port	Port number for the service	80
timestamp	Date and time the information was collected	2014-01-15T05:49:56.283713
hash	Numeric hash of the *data* property	
hostnames	List of hostnames for the IP	[“shodan.io”, “www.shodan.io”]
domains	List of domains for the IP	[“shodan.io”]
link	Network link type	Ethernet or modem
location	Geographic location of the device	*see below*
opts	Supplemental/ experimental data not contained in main banner	
org	Organization that is assigned the IP	Google Inc.
isp	ISP that is responsible for the IP space	Verizon Wireless
os	Operating system	Linux
uptime	Uptime of the IP in minutes	50
tags	List of tags that describe the purpose of the device (Enterprise-only)	[“ics”, “vpn”]
transport	Type of transport protocol used to collect banner; either “udp” or “tcp”	tcp

Elastic Properties

The following properties are collected for Elastic (formerly ElasticSearch):

[31] https://developer.shodan.io/api/banner-specification

Name	Description
elastic.nodes	List of nodes/ peers for the cluster and their information

HTTP(S) Properties

Shodan follows redirects of HTTP responses and stores all intermediate data in the banner. The only time the crawlers don't follow a redirect is if a HTTP request gets redirected to a HTTPS location and vice versa.

Name	Description
http.host	Hostname sent to grab the website HTML
http.html	HTML content of the website
http.html_hash	Numeric hash of the *http.html* property
http.location	Location of the final HTML response
http.redirects	List of redirects that were followed. Each redirect item has 3 properties: *host*, *data* and *location*.
http.robots	robots.txt file for the website
http.server	*Server* header from the HTTP response
http.sitemap	Sitemap XML for the website
http.title	Title of the website

Location Properties

The following properties are sub-properties of the **location** property that is at the top-level of the banner record.

Name	Description
area_code	Area code of the device's location
city	Name of the city
country_code	2-letter country code
country_code3	3-letter country code
country_name	Full name of the country
dma_code	Designated market area code (US-only)
latitude	Latitude
longitude	Longitude
postal_code	Postal code
region_code	Region code

SSH Properties

Name	Description
ssh.cipher	Cipher used during negotiation
ssh.fingerprint	Fingerprint for the device
ssh.kex	List of key exchange algorithms that are supported by the server
ssh.key	SSH key of the server
ssh.mac	Message authentication code algorithm

SSL Properties

If the service is wrapped in SSL then Shodan performs additional testing and makes the results available in the following properties:

Name	Description
ssl.acceptable_cas	List of certificate authorities that the server accepts
ssl.cert	Parsed SSL certificate
ssl.cipher	Preferred cipher for the SSL connection
ssl.chain	List of SSL certificates from the user certificate up to the root certificate
ssl.dhparams	Diffie-Hellman parameters
ssl.tlsext	List of TLS extensions that the server supports
ssl.versions	Supported SSL versions; if the value starts with a "-" then the service does *not* support that version (ex. "-SSLv2" means the service doesn't support SSLv2)

ISAKMP Properties

The following properties are collected for VPNs using the ISAKMP protocol (such as IKE):

Name	Description
isakmp.initiator_spi	Hex-encoded security parameter index for the initiator
isakmp.responder_spi	Hex-encoded security parameter index for the responder
isakmp.next_payload	The next paylod sent after the initiation
isakmp.version	Protocol version; ex "1.0"
isakmp.exchange_type	Exchange t ype
isakmp.flags.encryption	Encryption bit set: true or false
isakmp.flags.commit	Commit bit set: true or false
isakmp.flags.authentication	Authentication bit set: true or false

Name	Description
isakmp.msg_id	Hex-encoded ID for the message
isakmp.length	Size of the ISAKMP packet

Special Properties

_shodan

The **_shodan** property contains information about how the data was gathered by Shodan. It is different than al the other properties because it doesn't provide information about the device. Instead, it will tell you which banner grabber Shodan was using to talk to the IP. This can be important to understand for ports where multiple services might be operating on. For example, port 80 is most well-known for web servers but it's also used by various malware to circumvent firewall rules. The **_shodan** property would let you know whether the **http** module was used to collect the data or whether a malware module was used.

Name	Description
_shodan.crawler	Unique ID that identifies the Shodan crawler
_shodan.id	Unique ID for this banner
_shodan.module	Name of the Shodan module used by the crawler to collect the banner
_shodan.options	Configuration options used during the data collection
_shodan.hostname	Hostname to use when sending web requests
_shodan.options.referrer	Unique ID of the banner that triggered the scan for this port/ service

Example


```
{
       "timestamp": "2014-01-16T08:37:40.081917",
       "hostnames": [
          "99-46-189-78.lightspeed.tukrga.sbcglobal.net"
       ],
       "org": "AT&T U-verse",
       "guid": "1664007502:75a821e2-7e89-11e3-8080-808080808080",
       "data": "NTP\nxxxx.xxx.xxx.xxx:7546\n68.94.157.2:123\n68.94.156.17:123",
       "port": 123,
       "isp": "AT&T U-verse",
       "asn": "AS7018",
       "location": {
```

```
            "country_code3": "USA",
            "city": "Atlanta",
            "postal_code": "30328",
            "longitude": -84.3972,
            "country_code": "US",
            "latitude": 33.933500000000001,
            "country_name": "United States",
            "area_code": 404,
            "dma_code": 524,
            "region_code": null
        },
        "ip": 1664007502,
        "domains": [
            "sbcglobal.net"
        ],
        "ip_str": "99.46.189.78",
        "os": null,
        "opts": {
            "raw": "\\x97\\x00\\x03*\\x00\\x03\\x00H\\x00\\x00\\x00\\x00\\x00\\x00\
\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x01G\\x06\\xa7\\x8ec.\\xbdN\\x00\\
\x00\\x00\\x01\\x1dz\\x07\\x02\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\
\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\
\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\
\\x00q\\x00\\x00\\x00i\\x00\\x00\\x00\\x00\\x00\\x00\\x00XD^\\x9d\\x02c.\\xbdN\\\
x00\\x00\\x00\\x01\\x00{\\x04\\x04\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\
\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\
\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\
\x00\\x00q\\x00\\x00\\x00o\\x00\\x00\\x00\\x00\\x00\\x00\\x00YD^\\x9c\\x11c.\\xb\
dN\\x00\\x00\\x00\\x01\\x00{\\x04\\x04\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\\
x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\\
x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00",
            "ntp": {
                "more": false
            }
        }
    }
```

Appendix B: List of Search Filters

General Filters

Name	Description	Type
after	Only show results after the given date (dd/mm/yyyy)	string
asn	Autonomous system number	string
before	Only show results before the given date (dd/mm/yyyy)	string
category	Available categories: ics, malware	string
city	Name of the city	string
country	2-letter country code	string
geo	Accepts between 2 and 4 parameters. If 2 parameters: latitude,longitude. If 3 parameters: latitude,longitude,range. If 4 parameters: top left latitude, top left longitude, bottom right latitude, bottom right longitude.	string
hash	Hash of the **data** property	integer
has_ipv6	True/ False	boolean
has_screenshot	True/ False	boolean
hostname	Full hostname for the device	string
ip	Alias for **net** filter	string
isp	ISP managing the netblock	string
net	Network range in CIDR notation (ex. 199.4.1.0/24)	string
org	Organization assigned the netblock	string
os	Operating system	string
port	Port number for the service	integer
postal	Postal code (US-only)	string
product	Name of the software/ product providing the banner	string
region	Name of the region/ state	string
state	Alias for **region**	string
version	Version for the **product**	string
vuln	CVE ID for a vulnerability	string

HTTP Filters

Name	Description	Type
http.component	Name of web technology used on the website	string
http.component_category	Category of web components used on the website	string
http.html	HTML of web banners	string
http.html_hash	Hash of the website HTML	integer
http.status	Response status code	integer
http.title	Title for the web banner’s website	string

NTP Filters

Name	Description	
ntp.ip	IP addresses returned by monlist	string
ntp.ip_count	Number of IPs returned by initial monlist	integer
ntp.more	True/ False; whether there are more IP addresses to be gathered from monlist	boolean
ntp.port	Port used by IP addresses in monlist	integer

SSL Filters

Name	Description	Type
has_ssl	True/ False	boolean
ssl	Search all SSL data	string
ssl.alpn	Application layer protocols such as HTTP/2 (“h2”)	string
ssl.chain_count	Number of certificates in the chain	integer
ssl.version	Possible values: SSLv2, SSLv3, TLSv1, TLSv1.1, TLSv1.2	string
ssl.cert.alg	Certificate algorithm	string
ssl.cert.expired	True/ False	boolean
ssl.cert.extension	Names of extensions in the certificate	string
ssl.cert.serial	Serial number as an integer or hexadecimal string	integer/ string
ssl.cert.pubkey.bits	Number of bits in the public key	integer
ssl.cert.pubkey.type	Public key type	string
ssl.cipher.version	SSL version of the preferred cipher	string
ssl.cipher.bits	Number of bits in the preferred cipher	integer
ssl.cipher.name	Name of the preferred cipher	string

Telnet Filters

Name	Description	Type
telnet.option	Search all the options	string
telnet.do	The server requests the client do support these options	string
telnet.dont	The server requests the client to not support these options	string
telnet.will	The server supports these options	string
telnet.wont	The server doesn’t support these options	string

Appendix C: Search Facets

General Facets

Name	Description
asn	Autonomous system number
city	Full name of the city
country	Full name of the country
domain	Domain(s) for the device
has_screenshot	Has screenshot available
isp	ISP managing the netblock
link	Type of network connection
org	Organization owning the netblock
os	Operating system
port	Port number for the service
postal	Postal code
product	Name of the software/ product for the banner
region	Name of the region/ state
state	Alias for **region**
uptime	Time in seconds that the host has been up
version	Version of the **product**
vuln	CVE ID for vulnerability

HTTP Facets

Name	Description	
http.component	Name of web technology used on the website	string
http.component_category	Category of web components used on the website	string
http.html_hash	Hash of the website HTML	integer
http.status	Response status code	integer

NTP Facets

Name	Description
ntp.ip	IP addresses returned by monlist
ntp.ip_count	Number of IPs returned by initial monlist
ntp.more	True/ False; whether there are more IP addresses to be gathered from monlist
ntp.port	Port used by IP addresses in monlist

SSH Facets

Name	Description
ssh.cipher	Name of the cipher
ssh.fingerprint	Fingerprint for the device
ssh.mac	Name of MAC algorithm used (ex: hmac-sha1)
ssh.type	Type of authentication key (ex: ssh-rsa)

SSL Facets

Name	Description
ssl.version	SSL version supported
ssl.alpn	Application layer protocols
ssl.chain_count	Number of certificates in the chain
ssl.cert.alg	Certificate algorithm
ssl.cert.expired	True/ False; certificate expired or not
ssl.cert.serial	Certificate serial number as integer
ssl.cert.extension	Name of certificate extensions
ssl.cert.pubkey.bits	Number of bits in the public key
ssl.cert.pubkey	Name of the public key type
ssl.cipher.bits	Number of bits in the preferred cipher
ssl.cipher.name	Name of the preferred cipher
ssl.cipher.version	SSL version of the preferred cipher

Telnet Facets

Name	Description
telnet.option	Show all options
telnet.do	The server requests the client do support these options
telnet.dont	The server requests the client to not support these options
telnet.will	The server supports these options
telnet.wont	The server doesn’t support these options

Appendix D: List of Ports

Port	Service(s)
7	Echo
11	Systat
13	Daytime
15	Netstat
17	Quote of the day
19	Character generator
21	FTP
22	SSH
23	Telnet
25	SMTP
26	SSH
37	rdate
49	TACACS+
53	DNS
67	DHCP
69	TFTP, BitTorrent
70	Gopher
79	Finger
80	HTTP, malware
81	HTTP, malware
82	HTTP, malware
83	HTTP
84	HTTP
88	Kerberos
102	Siemens S7
110	POP3
111	Portmapper
119	NNTP
123	NTP
129	Password generator protocol
137	NetBIOS
143	IMAP
161	SNMP
175	IBM Network Job Entry
179	BGP
195	TA14-353a
311	OS X Server Manager
389	LDAP

Port	Service(s)
443	HTTPS
444	TA14-353a, Dell SonicWALL
445	SMB
465	SMTPS
500	IKE (VPN)
502	Modbus
503	Modbus
515	Line Printer Daemon
520	RIP
523	IBM DB2
554	RTSP
587	SMTP mail submission
623	IPMI
626	OS X serialnumbered
636	LDAPS
666	Telnet
771	Realport
789	Redlion Crimson3
873	rsync
902	VMWare authentication
992	Telnet (secure)
993	IMAP with SSL
995	POP3 with SSL
1010	malware
1023	Telnet
1025	Kamstrup
1099	Java RMI
1177	malware
1200	Codesys
1234	udpxy
1434	MS-SQL monitor
1521	Oracle TNS
1604	Citrix, malware
1723	PPTP
1741	CiscoWorks
1833	MQTT
1900	UPnP
1911	Niagara Fox
1962	PCworx
1991	malware
2000	iKettle, MikroTik bandwidth test
2082	cPanel
2083	cPanel
2086	WHM

Port	Service(s)
2087	WHM
2123	GTPv1
2152	GTPv1
2181	Apache Zookeeper
2222	SSH, PLC5, EtherNet/IP
2323	Telnet
2332	Sierra wireless (Telnet)
2375	Docker
2376	Docker
2404	IEC-104
2455	CoDeSys
2480	OrientDB
2628	Dictionary
3000	ntop
3306	MySQL
3310	ClamAV
3386	GTPv1
3388	RDP
3389	RDP
3460	malware
3541	PBX GUI
3542	PBX GUI
3689	DACP
3780	Metasploit
3787	Ventrilo
4000	malware
4022	udpxy
4040	Deprecated Chef web interface
4063	ZeroC Glacier2
4064	ZeroC Glacier2 with SSL
4369	EPMD
4443	Symantec Data Center Security
4444	malware
4500	IKE NAT-T (VPN)
4567	Modem web interface
4070	HID VertX/ Edge door controller
4800	Moxa Nport
4911	Niagara Fox with SSL
4949	Munin
5006	MELSEC-Q
5007	MELSEC-Q
5008	NetMobility
5009	Apple Airport Administration
5060	SIP

Port	Service(s)
5094	HART-IP
5222	XMPP
5269	XMPP Server-to-Server
5353	mDNS
5357	Microsoft-HTTPAPI/2.0
5432	PostgreSQL
5577	Flux LED
5632	PCAnywhere
5672	RabbitMQ
5900	VNC
5901	VNC
5938	TeamViewer
5984	CouchDB
6000	X11
6379	Redis
6666	Voldemort database, malware
6667	IRC
6881	BitTorrent DHT
6969	TFTP, BitTorrent
7218	Sierra wireless (Telnet)
7474	Neo4j database
7548	CWMP (HTTPS)
7777	Oracle
7779	Dell Service Tag API
8010	Intelbras DVR
8060	Roku web interface
8069	OpenERP
8087	Riak
8090	Insteon HUB
8099	Yahoo SmartTV
8112	Deluge (HTTP)
8139	Puppet agent
8140	Puppet master
8181	GlassFish Server (HTTPS)
8333	Bitcoin
8334	Bitcoin node dashboard (HTTP)
8443	HTTPS
8554	RTSP
8880	Websphere SOAP
8888	HTTP, Andromouse
8889	SmartThings Remote Access
9001	Tor OR
9002	Tor OR
9051	Tor Control

Port	Service(s)
9100	Printer Job Language
9151	Tor Control
9160	Apache Cassandra
9191	Sierra wireless (HTTP)
9418	Git
9443	Sierra wireless (HTTPS)
9595	LANDesk Management Agent
9600	OMRON
9869	OpenNebula
9009	Julia
10001	Automated Tank Gauge
10243	Microsoft-HTTPAPI/2.0
11211	Memcache
17000	Bose SoundTouch
17185	VxWorks WDBRPC
12345	Sierra wireless (Telnet)
13579	Media player classic web interface
14147	Filezilla FTP
16010	Apache Hbase
18245	General Electric SRTP
20000	DNP3
20547	ProconOS
21025	Starbound
21379	Matrikon OPC
23023	Telnet
23424	Serviio
25105	Insteon Hub
25565	Minecraft
27015	Steam A2S server query, Steam RCon
27017	MongoDB
28017	MongoDB (HTTP)
30313	Gardasoft Lighting
30718	Lantronix Setup
32400	Plex
37777	Dahuva DVR
44818	EtherNet/IP
47808	Bacnet
49152	Supermicro (HTTP)
49153	WeMo Link
50070	HDFS Namenode
51106	Deluge (HTTP)
53413	Netis backdoor
54138	Toshiba PoS
55553	Metasploit

Port	Service(s)
55554	Metasploit
62078	Apple iDevice
64738	Mumble

Appendix E: Sample SSL Banner


```
{
    "hostnames": [],
    "title": "",
    "ip": 2928565374,
    "isp": "iWeb Technologies",
    "transport": "tcp",
    "data": "HTTP/1.1 200 OK\r\nExpires: Sat, 26 Mar 2016 11:56:36 GMT\r\nExpire\
s: Fri, 28 May 1999 00:00:00 GMT\r\nCache-Control: max-age=2592000\r\nCache-Cont\
rol: no-store, no-cache, must-revalidate\r\nCache-Control: post-check=0, pre-che\
ck=0\r\nLast-Modified: Thu, 25 Feb 2016 11:56:36 GMT\r\nPragma: no-cache\r\nP3P:\
 CP=\"NON COR CURa ADMa OUR NOR UNI COM NAV STA\"\r\nContent-type: text/html\r\n\
Transfer-Encoding: chunked\r\nDate: Thu, 25 Feb 2016 11:56:36 GMT\r\nServer: sw-\
cp-server\r\n\r\n",
    "asn": "AS32613",
    "port": 8443,
    "ssl": {
        "chain": ["-----BEGIN CERTIFICATE-----\nMIIDszCCApsCBFBTb4swDQYJKoZIhvcN\
AQEFBQAwgZ0xCzAJBgNVBAYTAlVTMREw\nDwYDVQQIEwhWaXJnaW5pYTEQMA4GA1UEBxMHSGVybmRvbj\
ESMBAGA1UEChMJUGFy\nYWxsZWxzMRgwFgYDVQQLEw9QYXJhbGxlbHMgUGFuZWwxGDAWBgNVBAMTD1Bh\
cmFs\nbGVscyBQYW5lbDEhMB8GCSqGSIb3DQEJARYSaW5mb0BwYXJhbGxlbHMuY29tMB4X\nDTEyMDkx\
NDE3NTUyM1oXDTEzMDkxNDE3NTUyM1owgZ0xCzAJBgNVBAYTAlVTMREw\nDwYDVQQIEwhWaXJnaW5pYT\
EQMA4GA1UEBxMHSGVybmRvbjESMBAGA1UEChMJUGFy\nYWxsZWxzMRgwFgYDVQQLEw9QYXJhbGxlbHMg\
UGFuZWwxGDAWBgNVBAMTD1BhcmFs\nbGVscyBQYW5lbDEhMB8GCSqGSIb3DQEJARYSaW5mb0BwYXJhbG\
xlbHMuY29tMIIB\nIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAxc9Vy/qajKtFFnHxGOFPHTxm\
\nSOnsffWBTBfyXnK3h8u041VxvZDh3XkpA+ptg2fWOuIT0TTYuqw+tqiDmg8YTsHy\njcpMFBtXV2cV\
dhKXaS3YYlM7dP3gMmkGmH+ZvCgCYc7L9MIJxYJy6Zeuh67YxEMV\ngiU8mZpvc70Cg5WeW1uBCXtUAi\
jDLsVWnhsV3YuxlweEvkRpAk3EHehKbvgMnEZS\nQ30QySe0GAqC7bWzKrwsJAOUk/+Js18+3QKb/LmD\
a9cRjtFCTo6hYfPbfHj8RxQh\n4Xmnn/CtZ48wRQTqKXSO6+Zk3OuU7/jX1Gt/jxN6n77673e6uCsggT\
wut/EtNwID\nAQABMA0GCSqGSIb3DQEBBQUAA4IBAQBb/yTy76Ykwr7DBOPAXc766n73OsZizjAt\n1k\
mx7LxgN3X/wFxD53ir+sdOqbPgJl3edrE/ZG9dNl6LhUBbUK+9s6z9QicEfSxo\n4uQpFSywbGGmXInE\
ZmyT4SsOLi/hNgy68f49LO1h6rn/p7QgIKd31g7189ZfFkFb\nRdD49s1l/Cc5Nm4XapUVvmnS91MlPk\
/OOIg1Lu1rYkuc8sIoZdPbep52H3Ga7TjG\nkmO7nUIii0goB7TQ63mU67+NWHAmQQ8CtCDCN49kJyen\
1WFjD6Je2U4q0IFQrxHw\nMy+tquo/n/sa+NV8QOj1gMVcFsLhYm7Z5ZONg0QFXSAL+Eyj/AwZ\n----\
-END CERTIFICATE-----\n"],
        "cipher": {
            "version": "TLSv1/SSLv3",
            "bits": 256,
```

```
            "name": "DHE-RSA-AES256-GCM-SHA384"
        },
        "alpn": [],
        "dhparams": {
            "prime": "b10b8f96a080e01dde92de5eae5d54ec52c99fbcfb06a3c69a6a9dca52\
d23b616073e28675a23d189838ef1e2ee652c013ecb4aea906112324975c3cd49b83bfaccbdd7d90\
c4bd7098488e9c219a73724effd6fae5644738faa31a4ff55bccc0a151af5f0dc8b4bd45bf37df36\
5c1a65e68cfda76d4da708df1fb2bc2e4a4371",
            "public_key": "2e30a6e455730b2f24bdaf5986b9f0876068d4aa7a4e15c9a1b9c\
a05a420e8fd3b496f7781a9423d3475f0bedee83f0391aaa95a738c8f0e250a8869a86d41bdb0194\
66dba5c641e4b2b4b82db4cc2d4ea8d9804ec00514f30a4b6ce170b81c3e1ce4b3d17647c8e5b8f6\
65bb7f588100bcc9a447d34d728c3709fd8a5b7753b",
            "bits": 1024,
            "generator": "a4d1cbd5c3fd34126765a442efb99905f8104dd258ac507fd6406c\
ff14266d31266fea1e5c41564b777e690f5504f213160217b4b01b886a5e91547f9e2749f4d7fbd7\
d3b9a92ee1909d0d2263f80a76a6a24c087a091f531dbf0a0169b6a28ad662a4d18e73afa32d779d\
5918d08bc8858f4dcef97c2a24855e6eeb22b3b2e5",
            "fingerprint": "RFC5114/1024-bit MODP Group with 160-bit Prime Order\
 Subgroup"
        },
        "versions": ["TLSv1", "-SSLv2", "SSLv3", "TLSv1.1", "TLSv1.2"]
    },
    "html": "\n\t\t<html><head>\n\t\t<meta charset=\"utf-8\">\n\t\t<meta http-eq\
uiv=\"X-UA-Compatible\" content=\"IE=edge,chrome=1\">\n\t\t<title></title>\n\t\t\
<script language=\"javascript\" type=\"text/javascript\" src=\"/javascript/commo\
n.js?plesk_version=psa-11.0.9-110120608.16\"/></script>\n\t\t<script language=\"\
javascript\" type=\"text/javascript\" src=\"/javascript/prototype.js?plesk_versi\
on=psa-11.0.9-110120608.16\"></script>\n\t\t<script>\n\t\t\tvar opt_no_frames = \
false;\n\t\t\tvar opt_integrated_mode = false;\n\t\t</script>\n\t\t\n\t\t</head>\
<body onLoad=\";top.location='/login.php3?window_id=&amp;requested_url=https%3A%\
2F%2F174.142.92.126%3A8443%2F';\"></body><noscript>You will be redirected to the\
 new address in 15 seconds... If you are not automatically taken to the new loca\
tion, please enable javascript or click the hyperlink <a href=\"/login.php3?wind\
ow_id=&amp;requested_url=https%3A%2F%2F174.142.92.126%3A8443%2F\" target=\"top\"\
>/login.php3?window_id=&amp;requested_url=https%3A%2F%2F174.142.92.126%3A8443%2F\
</a>.</noscript></html><!--_____________________________________________________\
________________________________________________________________________________\
________________________________________________________________________________\
_________________________IE error page size limitation__________________________\
________________________________________________________________________________\
________________________________________________________________________________\
____________________________________________________-->",
```

```
    "location": {
        "city": null,
        "region_code": "QC",
        "area_code": null,
        "longitude": -73.5833,
        "country_code3": "CAN",
        "latitude": 45.5,
        "postal_code": "H3G",
        "dma_code": null,
        "country_code": "CA",
        "country_name": "Canada"
    },
    "timestamp": "2016-02-25T11:56:52.548187",
    "domains": [],
    "org": "iWeb Technologies",
    "os": null,
    "_shodan": {
        "options": {},
        "module": "https",
        "crawler": "122dd688b363c3b45b0e7582622da1e725444808"
    },
    "opts": {
        "heartbleed": "2016/02/25 03:56:45 ([]uint8) {\n 00000000  02 00 74 63 6\
5 6e 73 75  73 2e 73 68 6f 64 61 6e  |..tcensus.shodan|\n 00000010  2e 69 6f 53 \
45 43 55 52  49 54 59 20 53 55 52 56  |.ioSECURITY SURV|\n 00000020  45 59 fe 7a\
 a2 0d fa ed  93 42 ed 18 b0 15 7d 6e  |EY.z.....B....}n|\n 00000030  29 08 f6 f\
8 ce 00 b1 94  b5 4b 47 ac dd 18 aa b9  |)........KG.....|\n 00000040  db 1c 01 \
45 95 10 e0 a2  43 fe 8e ac 88 2f e8 75  |...E....C..../.u|\n 00000050  8b 19 5f\
 8c e0 8a 80 61  56 3c 68 0f e1 1f 73 9e  |.._....aV<h...s.|\n 00000060  61 4f d\
a db 90 ce 84 e3  79 5f 9d 6c a0 90 ff fa  |aO......y_.l....|\n 00000070  d8 16 \
e8 76 07 b2 e5 5e  8e 3e a4 45 61 2f 6a 2d  |...v...^.>.Ea/j-|\n 00000080  5d 11\
 74 94 03 3c 5d                              |].t..<]|\n}\n\n2016/02/25 03:56:45\
 174.142.92.126:8443 - VULNERABLE\n",
        "vulns": ["CVE-2014-0160"]
    },
    "ip_str": "174.142.92.126"
}
```

Exercise Solutions

Website

Exercise 1

```
title:4sics
```

Exercise 2

```
rfb authentication disabled
```

Exercise 3

```
vuln:CVE-2014-0160 country:se ssl.version:sslv3
```

```
vuln:CVE-2014-0160 org:"your organization"
```

Exercise 4

```
category:ics city:"your city name"
```

Exercise 5

```
category:malware country:us
```

Command-Line Interface

Exercise 1

```
shodan download --limit -1 heartbleed-results country:se,no vuln:CVE-2014-0160
shodan parse --filters location.country_code:SE -O heartbleed-sweden heartbleed-\
results.json.gz
```

> Note: The –**filters** argument does case-sensitive searching on properties that are strings, hence the Swedish country code has to be upper-case.

Exercise 2

```
mkdir data
shodan stream --limit 1000 --datadir data/
shodan convert data/* kml

# Upload the KML file to https://www.google.com/maps/d/
```

Exercise 3

```
#!/bin/bash

shodan download --limit -1 malware.json.gz category:malware

for ip in `shodan parse --fields ip_str malware.json.gz`
do
        iptables -A OUTPUT -d $ip -j DROP
done
```

Shodan API

Replace *YOUR_API_KEY* with the API key for your account as seen on your Shodan Account website[32].

Exercise 1

```
#!/usr/bin/env python

# Initialize Shodan
import shodan
api = shodan.Shodan("YOUR_API_KEY")

# Create a new alert
alert = api.create_alert('My first alert', '198.20.69.0/24')

try:
        # Subscribe to data for the created alert
        for banner in api.stream.alert(alert['id']):
                print banner
except:
        # Cleanup if any error occurs
        api.delete_alert(alert['id'])
```

Tip: Here is a solution using the Shodan command-line interface's **alert** commands:

[32] https://account.shodan.io

```
# Create the alert
shodan alert create "My first alert" 198.20.69.0/24

# Subscribe to the real-time feed and store the data in the "/tmp" directory
shodan stream --alerts=all --datadir=/tmp

# Once we're done remove all alerts
shodan alert clear
```

Exercise 2

```
mkdir images
```

Run the above command to generate a directory to store the images in. Then save the following code in a file such as `image-stream.py`:

```python
#!/usr/bin/env python

import shodan

output_folder = 'images/'
api = shodan.Shodan("YOUR_API_KEY")

for banner in api.stream.banners():
        if 'opts' in banner and 'screenshot' in banner['opts']:
                # All the images are JPGs for now
                # TODO: Use the mimetype to determine file extension
                # TODO: Support IPv6 results

                # Create the file name using its IP address
                filename = '{}/{}.jpg'.format(output_folder, banner['ip_str'])

                # Create the file itself
                output = open(filename, 'w')

                # The images are encoded using base64
                output.write(banner['opts']['screenshot']['data'].decode('base64'))
```