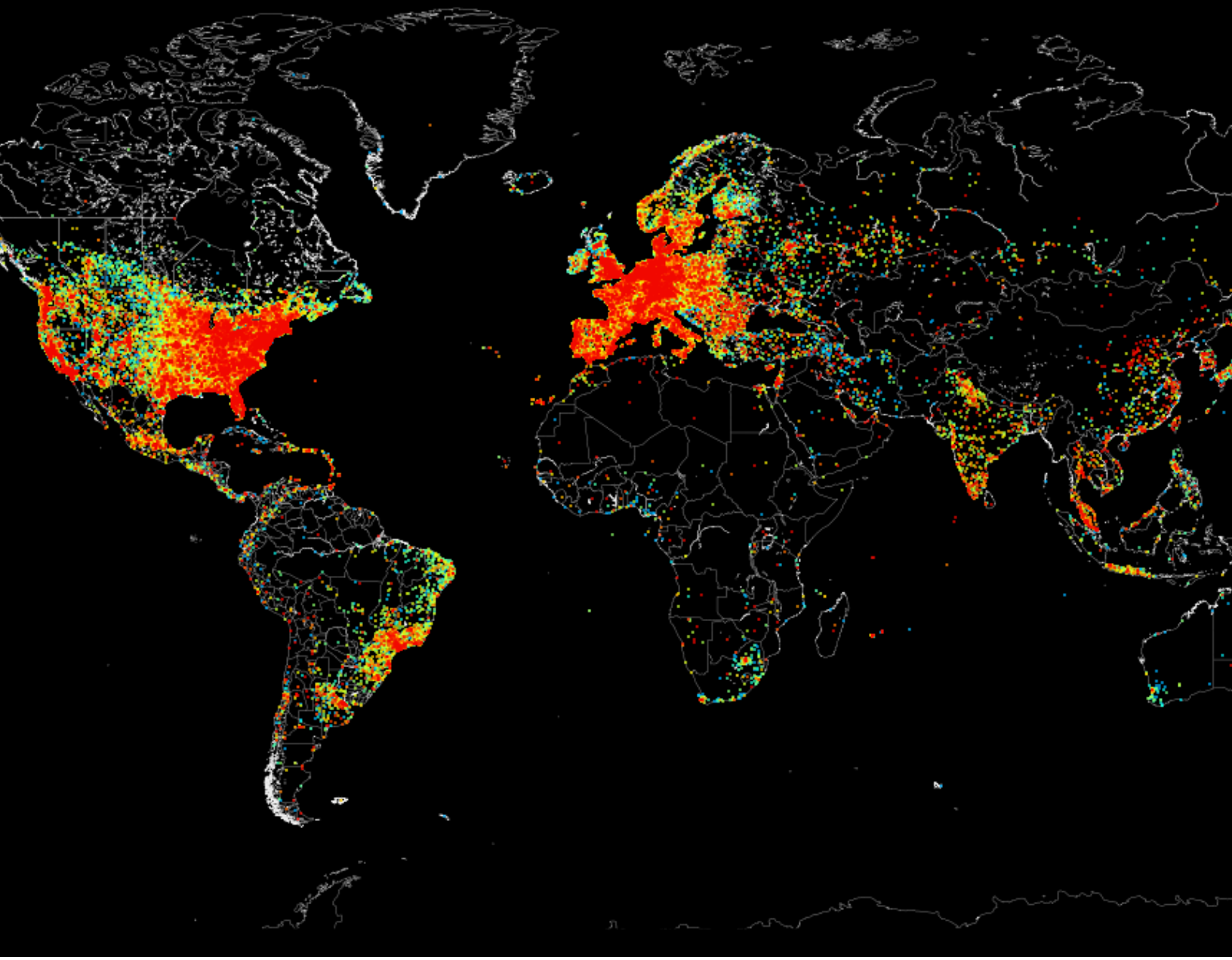


Shodan Enterprise Data License

Collect. Analyze. Visualize. Unlock Internet Intelligence for your Company.

Table of Contents

Introduction

The Shodan Enterprise Data License provides full, unlimited access to all data collected by the Shodan crawlers as well as unrestricted access to the Shodan website for the entire company. At the moment, the Shodan crawlers check for more than 260 different ports/ services and collect on average 500-600 banners every second.

260+ Ports	**500 – 600** banners/ s	**1+ Billion** banners/ month

The crawlers are located around the world (USA, China, Iceland, France and more) to ensure the data collected isn't geographically biased. This means that you get a truly global view of the publicly-accessible devices on the Internet.

Bulk Database Updates

All the data that Shodan collects is made available through a variety of methods: Direct Download, Real-Time Stream and optional hard drive delivery. All the below methods include all services that Shodan crawls for, including ones that haven't been officially announced or supported on the website.

Direct Download

Download all the data that was gathered for a day from a private, secured web location. The location is the central repository for all Shodan-related data and the size of a daily download is between 40-50 GB of compressed JSON; uncompressed it is between 200-300 GB of data per day.

Real-Time Stream

The Streaming API (https://developer.shodan.io/api/stream) provides a real-time feed of all the data that the Shodan crawlers are gathering. This is the primary method for consuming the Shodan Enterprise data, with the Direct Downloads being available in case you miss a day or prefer to do daily batch processing. The stream averages between 500-600 banners per second.

Hard Drive Delivery

If the other options aren't to your liking, Shodan also offers the option to send out a hard drive each month containing all the data gathered for the given month.

The website only contains a fraction of the data that is collected. Shodan grabs more than just banners, it often also stores additional service information (ex. the list of peers for bitcoin, SSL cert for HTTPS or robots.txt for HTTP). The official website also doesn't include the latest research as it takes time to integrate it into the website but is immediately available to data license customers.

IPv6 Ready

The entire Shodan infrastructure is IPv6-enabled which means that all features listed in this document (SSL testing, on-demand scanning, screenshots etc.) are available for both IPv4 and IPv6 devices.

Unlimited Access for Everybody

Unlimited Website Access

The Shodan Data License includes account upgrades for all employees of the company. An upgraded account has access to all features of the website, including Shodan Maps, Shodan Images, extended search results and Telnet/ HTTPS services.

Unlimited API Access

In addition to providing unlimited and unrestricted access to the Shodan website, the Enterprise Data License also provides all employees of the company with unlimited programmatic access to Shodan via its REST and Streaming API. For information on the methods available in the API, please visit the documentation at:

https://developer.shodan.io/api

```
Submitting Internet scan to Shodan...Done
Saving results to file: 9042-http.json.gz
Waiting for data, please stand by...
218.199.188.0                   9042
207.62.175.0                    9042
180.175.190.0                   9042
14.114.193.0                    9042
218.199.183.0                   9042
117.41.183.0                    9042
117.41.185.0                    9042
117.41.187.0                    9042
117.41.184.0                    9042
117.41.182.0                    9042
198.248.79.1                    9042
223.154.121.1                   9042
82.127.121.1                    9042        LPuteaux-656-1-244-1.w82-127.abo.wanadoo.fr
152.163.29.1                    9042        bos-1006a.blue.aol.com
5.154.11.1                      9042
188.18.12.1                     9042
198.248.77.0                    9042
194.9.35.1                      9042
198.248.75.0                    9042
198.248.76.0                    9042
198.248.78.0                    9042
198.248.79.0                    9042
107.21.93.1                     9042        ec2-107-21-93-1.compute-1.amazonaws.com
198.248.92.1                    9042
```

On-Demand Internet Scans

It is possible to submit IPs/ netblocks to the Shodan API for scanning. This is a direct line to the Shodan infrastructure, effectively giving you the option to scan the Internet using the Shodan crawlers. Checkout the **/shodan/scan** and **/shodan/scan/internet** documentation at:

https://developer.shodan.io/api

If you use the Python library for Shodan then you're already able to submit IPs using code such as:

```
import shodan
api = shodan.Shodan(YOUR_API_KEY)
api.scan(['199.20.33.0/24', '201.1.40.32'])
```

The above code submits the netblock **199.20.33.0/24** as well as the IP **201.1.40.32** to be scanned by Shodan. The following code submits an Internet-scan on port 8080 for web servers:

```
import shodan
api = shodan.Shodan(YOUR_API_KEY)
api.scan_internet(8080, 'http')
```

The API also supports scanning custom ports. The following code scans the IP 198.20.59.84 for Modbus on port 501 which is a non-standard port for Modbus (the default port is 502):

```
import shodan
api = shodan.Shodan(YOUR_API_KEY)
api.scan({
    '198.20.59.84': [
        (501, 'modbus')
    ]
})
```

The results for the scans will be available in the real-time stream as well as the daily downloads.

Website Components

The crawlers now 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 new **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 has been expanded with a new filter (**http.component**) and 2 new 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 you could use the following *shodan* command:

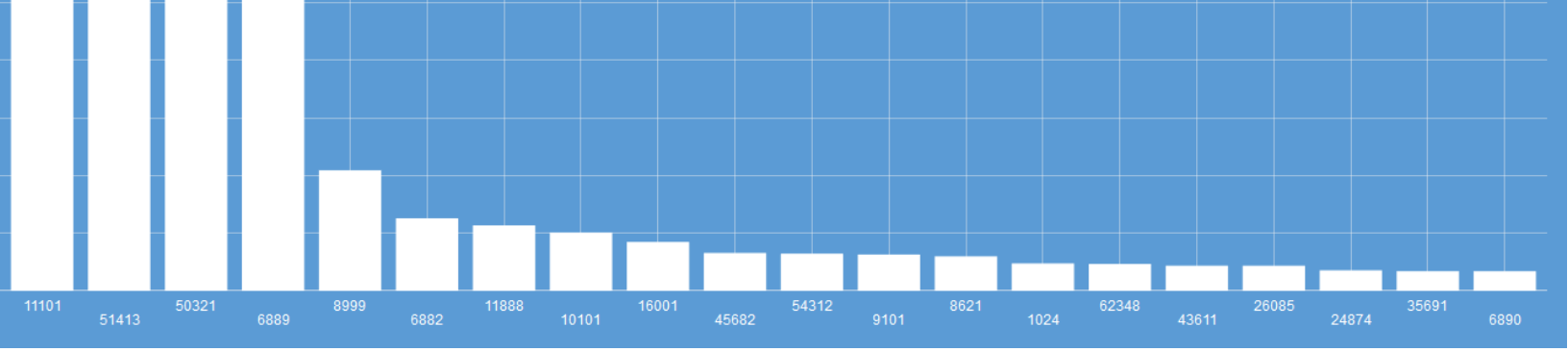
```
$ shodan stats --facets http.component_category:1000 http
Top 47 Results for Facet: http.component_category

javascript-frameworks          8,982,996

web-frameworks                 1,708,503

programming-languages          1,409,763

font-scripts                   1,280,397
...
```



Cascading Banners

If a banner returns information about peers or otherwise has information about another IP address that runs a service then the crawlers now 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
54.70.96.157    61770
85.82.92.188    42115
52.126.164.27   3784
212.18.46.72    6103
26.225.54.70    27775
243.11.233.190  30663
186.123.119.171 56163
80.5.61.111     28838
106.117.26.144  13894
111.77.25.250   2985
```

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.

Cascading is currently enabled for the following protocols:

- DHT
- Bitcoin
- HTTP ←→ HTTPS

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

- **_shodan.id**
- **_shodan.options.referrer**

_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. We are planning on making this a mandatory property that will exist on all banners.

_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.

SSL in Depth

SSL is becoming an increasingly 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.

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.

TLS Extensions

In addition to the SSL testing performed for all SSL services the crawlers now also store the list of TLS extensions that the server supports. The extensions are available in the **ssl.tlsext** property as a list of objects where each object has an **id** and a **name**. The names and ids are taken from IANA:

http://www.iana.org/assignments/tls-extensiontype-values/tls-extensiontype-values.xhtml

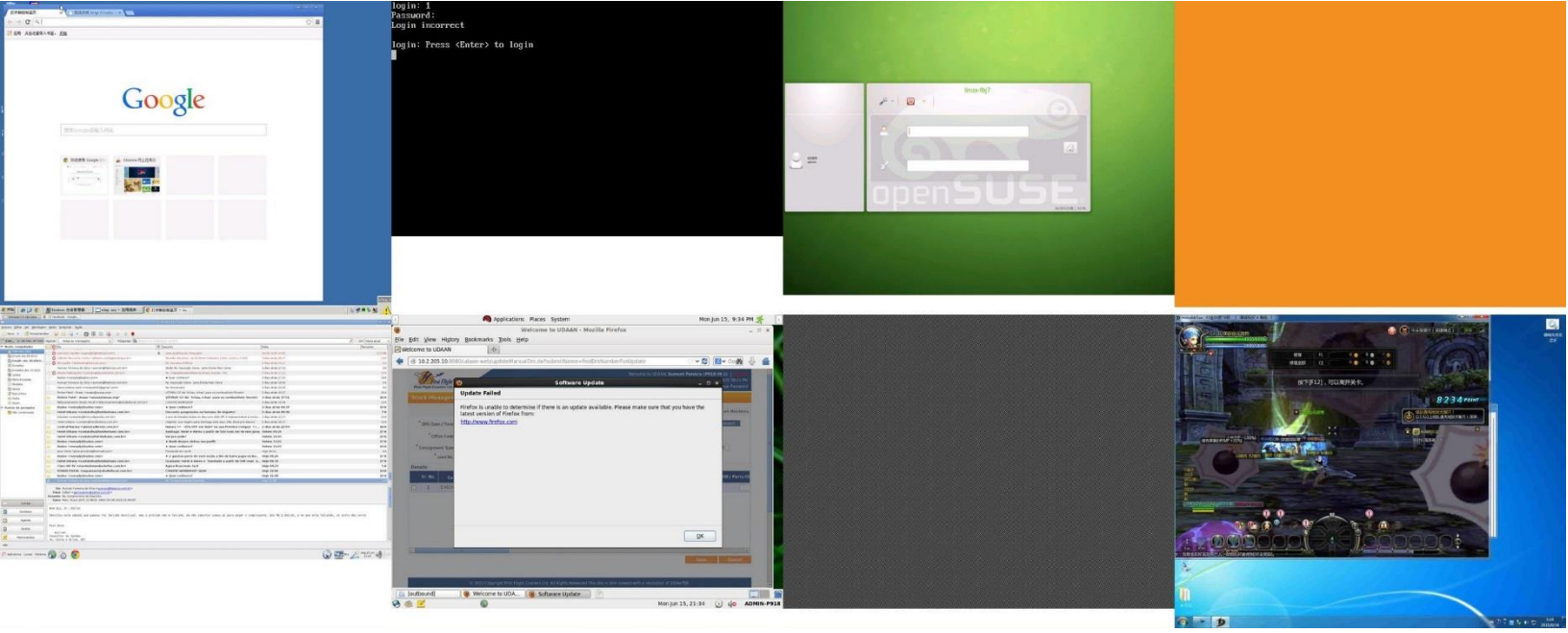
Below is a sample **ssl** property containing the new **ssl.tlsext** property alongside the other SSL information:

```
{
	"ssl": {
		"tlsext": [{
			"id": 65281,
			"name": "renegotiation_info",
		}, {
			"id": "35,
			"name": "SessionTicket"
		}]
	}
}
```

HTTP/2

HTTP/2.0 is the next version of the protocol powering websites and it promises many improvements over HTTP/1.x. There are a few different ways that a server can advertise support for HTTP/2.0 but the most common one is during the SSL handshake. We've added support in Shodan for tracking the negotiated HTTP versions and the data can be found in the **ssl.alpn** property.

```
{
	"ssl": {
		"alpn": ["h2", "http/1.1"]
	}
}
```



Screenshots

Sometimes a banner doesn't tell you much about the device or service itself. In those instances, the Shodan crawlers attempt to take a screenshot to help you learn more about the service. Image data is gathered from 5 different sources:

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

The results of the screenshot collection are stored in the **opts.screenshot** property, which has 2 properties:

- **data**: the image data converted to a base64-encoded string
- **mime**: the mimetype of the image, usually **image/jpeg**

Shodan also detects if a response from a non-standard port looks like a known protocol such as VNC and tries to grab a screenshot.