# Get reverse NS (aka passive DNS) records for a list of IPs in Python

Posted on June 30, 2021

Passive DNS introduced by Florian Weimer in 2005 is now a central resource in IP security investigations, security of the operation of the domain name system (DNS), and many more. A Passive DNS database contains observed events whenever an IP resolves to a domain name in a DNS communication. Hence, it is a database independent from the current state as well as the physical infrastructure of the DNS itself. In addition, it contains time information: the date and time when such a resolution was first and last observed; this cannot be found out from the DNS.

One of the easiest ways to obtain such data is by using WhoisXML API's services. In the present blog, we focus on the reverse lookup: using an IPv4 address we want to reveal the domain names that these IPs belonged to on certain dates.

We will be working in Python in a Linux or Mac OS X command-line but it is easy to do the same in a Windows command-line environment, too, provided that Python is functional. Our description is intended for beginners; only very basic Python and shell-script skills are assumed. We will use the reverse IP API and the supplied Python package that makes the use of the API very simple. This library can be installed with Python's package manager, pip. The following command, when run from the shell, will download and install it:

```
pip3 install reverse-ip
```

(Optionally you may want to do that as a root user to make the library available on your whole system, or do it in a Python virtual environment to limit the scope of the installation.)

The reverse-ip package is now technically ready to use but a subscription to the API is also needed, a free subscription is also available. Upon subscription, you will receive an API key, a string, which will be referred to as YOUR_API_KEY in what follows.

Having all prerequisites at hand we can now accomplish our task. So, let us assume that we have a list of IP addresses in a file named ips_demo.csv. For demonstration, we will use the following example:

```
172.67.155.63
104.21.20.75
```

These are IP addresses belonging to WhoisXML API's web services but they are also shared with other services by our web provider.

We will implement a script that reads IP addresses from standard input and writes the output in a comma-separated values (csv) format to the standard output. So, let us edit the script, get_reverse_ip_of_a_list.py with a suitable programmer's text editor. First, let's just read the IPs and repeat them on the standard output, and this initial code reads:

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

for line in sys.stdin.readlines():
    ip = line.strip()
    sys.stdout.write('"%s",'%ip)

    sys.stdout.write("\n")
```

We could opt for a more sophisticated way of reading our IPs but let's keep things simple. We just read the lines coming from the standard input and remove any additional characters like newlines with .strip(), and write the IP to the standard output, between double quotes followed by a comma; the results will follow here. Then we write a newline, since we have already kept the room between the two sys.stdout.write calls for the main part of the script.

But before extending it, let's give it a try: in the shell command-line let's do

```
chmod +x get_reverse_ip_of_a_list.py
get_reverse_ip_of_a_list.py < ips_demo.csv
```

resulting in the following output:

```
"172.67.155.63",
"104.21.20.75",
```

which is just as expected.

Let us now extend our script to do its job. It will read:

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

import datetime
from time import sleep
from reverseip import Client

reverseip = Client('YOUR_API_KEY')

for line in sys.stdin.readlines():
    ip = line.strip()
    sys.stdout.write('"%s",'%ip)

    data = reverseip.data(ip)
    for record in data['result']:
sys.stdout.write('"%s","%s","%s",'%(record['name'],
    datetime.datetime.fromtimestamp(
int(record['first_seen'])).strftime('%c'),
    datetime.datetime.fromtimestamp(
int(record['last_visit'])).strftime('%c')))
    sys.stdout.write("\n")
```

```
        sleep(.1)
```

Do not forget to replace YOUR_API_KEY with your actual API key to make it work. What we do is very simple: we just import the Client class from the module, and create the instance reverseip, where we pass the API key. Then the result will be obtained with the call of reverseip.data, readily in a Python dictionary; for the details of the output data format please consult the API documentation.

For the time being it is sufficient to know that the result field of the output, i.e., data['result'] is an iterator for the found records. (More precisely, for up to 300 records. In general, whenever the number of domain names for an IP is above 50 or 100, infrastructure is shared and possibly not so meaningful to have all the records. If you need them all, however, the API docs describe how to do so.)

So, we iterate through the records if any. Each record has 3 fields, name is the actual domain name, whereas first_seen and last_visit give the datetime of the first and last observation of this IP-name pair. This is provided by the API in the form of timestamp; we convert it into readable datetimes in the date format and the timezone of the local system by the seemingly cumbersome but logical functions of the datetime standard Python library. We append each triplet to the given output line.

Finally, we wait 0.1 seconds to avoid running into any throttling limit of the API. (The maximum number of requests per second is actually 30, so this may even be omitted.) Let's see what we have done:

```
./get_reverse_ip_of_a_list.py < ips_demo.csv
```

will now result in

```
"172.67.155.63","labbry.com","Fri Jul 26 07:03:05 2019","Fri May  7 11:09:0
"104.21.20.75","taalmedia.com","Fri Jun 21 13:50:38 2019","Fri Apr 23 18:12
```

at least at the time of writing the present blog. (We have truncated the second line for

typographical reasons.) You may want to direct the standard output to a file:

```
./get_reverse_ip_of_a_list.py < ips_demo.csv > ips_pdns.csv
```

so that ips_pdns.csv can be imported with your favorite office spreadsheet.

In conclusion, we have demonstrated how to obtain passive DNS data very easily in Python, even if you have just started to work with Python you can quickly get to it. Meanwhile, for the advanced Python programmer the message is simply that after installing the reverse-ip Python package and getting in hold of an API key from WhoisXML API,

```
from reverseip import Client

reverseip = Client('YOUR_API_KEY')

data = reverseip.data(ip)
```

gives you a Python dictionary with passive DNS records for an IP address. And this is perhaps the easiest way to obtain such data with a single function call.