How to Port Scan a Website

infosecmatter.com/how-to-port-scan-a-website

February 5, 2020

Port scanning of a website may sound simple, but before firing up the Nmap command on our target, we should do a little digging first. Are we going to scan the right target? In this article we will look closer on how to properly port scan a website. All examples below were produced on Kali Linux.

General approach to port scanning a website

When we are talking about port scanning, we have to think in network layer in terms of IP addresses – not websites or hostnames.

Technically speaking, we cannot port scan a website or a hostname – we can only port scan an IP address, which the target hostname / FQDN (Fully Qualified Domain Name) (e.g. www.example.com) resolves to.

So with that having cleared out, the first thing we should do is to perform DNS lookup on the target.

Rule number 1: Resolve the target hostname to IP address(es)

We can use any of the following commands to do the job:

```
host www.example.com
dig www.example.com +short
getent ahosts www.example.com
```

Now after we obtained all IP addresses, we have to ensure that they truly belong to our target.

Rule number 2: Ensure that you are going to scan the right IP address(es)

The thing is that the target website may be using a CDN (Content Delivery Network) provider, effectively hiding its real IP address behind it.

Or it may be load balanced across multiple IP addresses. We will never know these things without doing a little digging. We will discuss this in the next section.

Rule number 3: Select the right port scan strategy

Depending on the nature of the test (white-box, grey-box, black-box, red-team etc.) we may have to use different port scanning strategies.

We cover this in detail in the final section of this article.

Why is it important to determine if website is using CDN?

If the target website uses CDN, it means that the website's real IP address is hidden behind the CDN. And the IP address(es) that we see when we are doing DNS queries are belonging to the CDN provider – not the actual target website!

This is of course crucial information. By scanning CDN's IP address, we would be obviously scanning something completely different than what we want to scan.

How to determine if website is using CDN?

There are multiple ways how to identify CDN. We can divide them into the following two categories:

- 3rd party online tool – This is the fast and easy way, but not always accurate.
- Manual way – This requires us to type a command or two, but it's always 100% accurate.

Let's break them down.

Method 1: Online tools to identify CDN

There are number of online tools that can help us determine whether a particular website is using CDN or not. Here are some of the free ones:

But as we indicated above, sometimes these 3rd party tools can give inaccurate information. Therefore, it is always good to verify everything manually to be 100% certain. It's easy to do as well, so why not to get used to it and do it always manually.

Method 2: Manual way to identify CDN

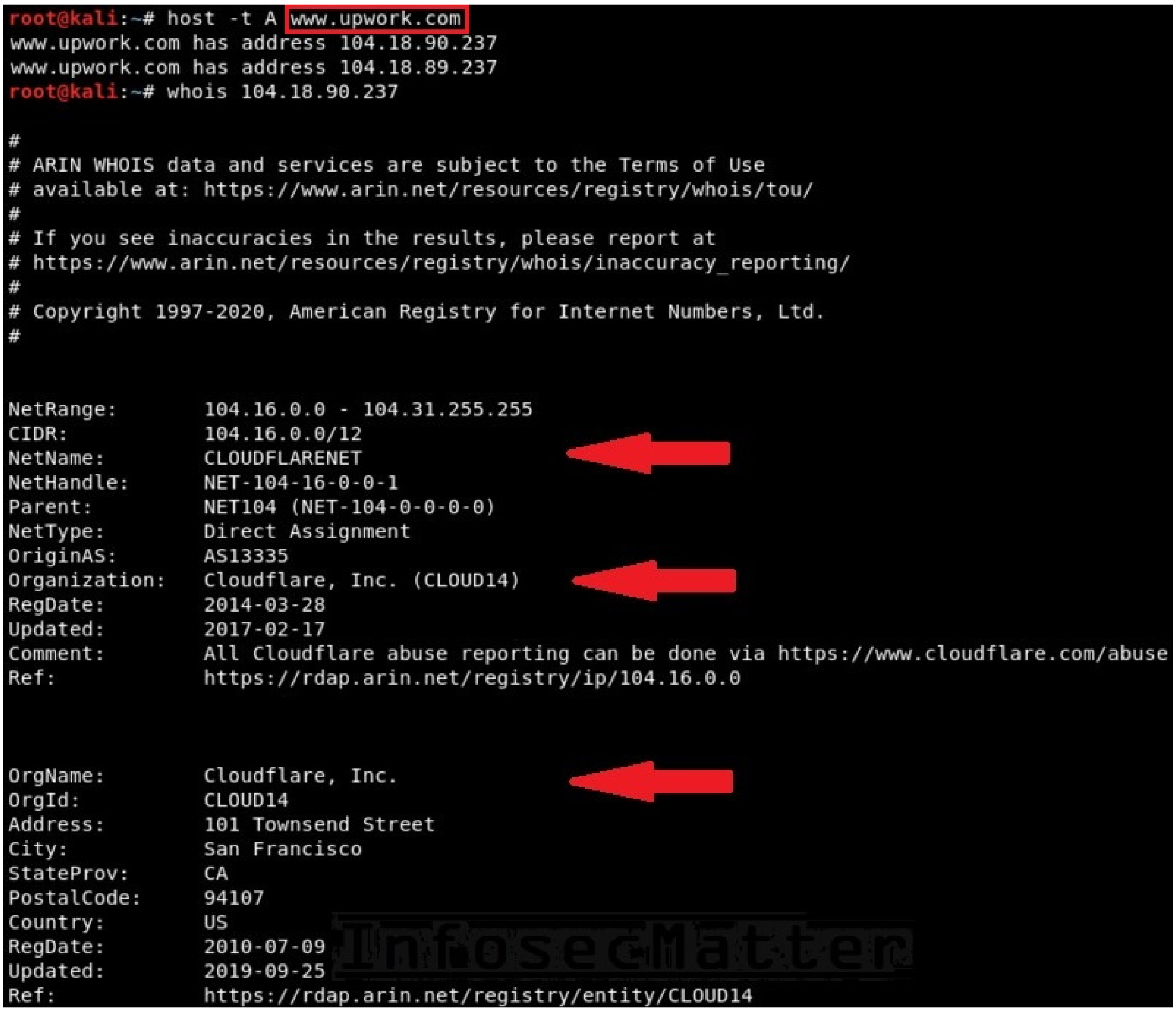
First we have to resolve the website's hostname to the IP address(es).

```
host -t A www.example.com
```

Then we have to look up who owns those IP addresses in the WHOIS database. This will give us 100% certainty.

```
whois <IP-ADDRESS>
```

If we are seeing evidences of CDN's (names like Amazon, Cloudflare, Azure etc.) all over the place, there are no doubts – the website is using CDN. Here's an example:



```
root@kali:~# host -t A www.upwork.com
www.upwork.com has address 104.18.90.237
www.upwork.com has address 104.18.89.237
root@kali:~# whois 104.18.90.237

#
# ARIN WHOIS data and services are subject to the Terms of Use
# available at: https://www.arin.net/resources/registry/whois/tou/
#
# If you see inaccuracies in the results, please report at
# https://www.arin.net/resources/registry/whois/inaccuracy_reporting/
#
# Copyright 1997-2020, American Registry for Internet Numbers, Ltd.
#


NetRange:       104.16.0.0 - 104.31.255.255
CIDR:           104.16.0.0/12
NetName:        CLOUDFLARENET
NetHandle:      NET-104-16-0-0-1
Parent:         NET104 (NET-104-0-0-0-0)
NetType:        Direct Assignment
OriginAS:       AS13335
Organization:   Cloudflare, Inc. (CLOUD14)
RegDate:        2014-03-28
Updated:        2017-02-17
Comment:        All Cloudflare abuse reporting can be done via https://www.cloudflare.com/abuse
Ref:            https://rdap.arin.net/registry/ip/104.16.0.0



OrgName:        Cloudflare, Inc.
OrgId:          CLOUD14
Address:        101 Townsend Street
City:           San Francisco
StateProv:      CA
PostalCode:     94107
Country:        US
RegDate:        2010-07-09
Updated:        2019-09-25
Ref:            https://rdap.arin.net/registry/entity/CLOUD14
```

The target website is using CDN, what now?

Well, it depends.

In a **white-box** test approach or in a typical VAPT (Vulnerability Assessment / Penetration Testing) scenario, the client would provide hostnames and IP addresses of all assets in scope as part of the scoping information, because the objective is typically to test the actual assets in scope.

In more **black-box** approach or in a **red team** exercise, we would have to find this out by ourselves. The client would not give us this information. Therefore, we would have to somehow disclose the real IP address of the website.

How to disclose website's real IP address behind CDN?

Good news is that it is almost always possible. On the other hand, it may require deeper digging and using a 3rd party services which are not always free. Disclosing real IP address behind CDN is a topic beyond the scope of this article, but let's briefly mention some of the techniques and methods that can be used to do this:

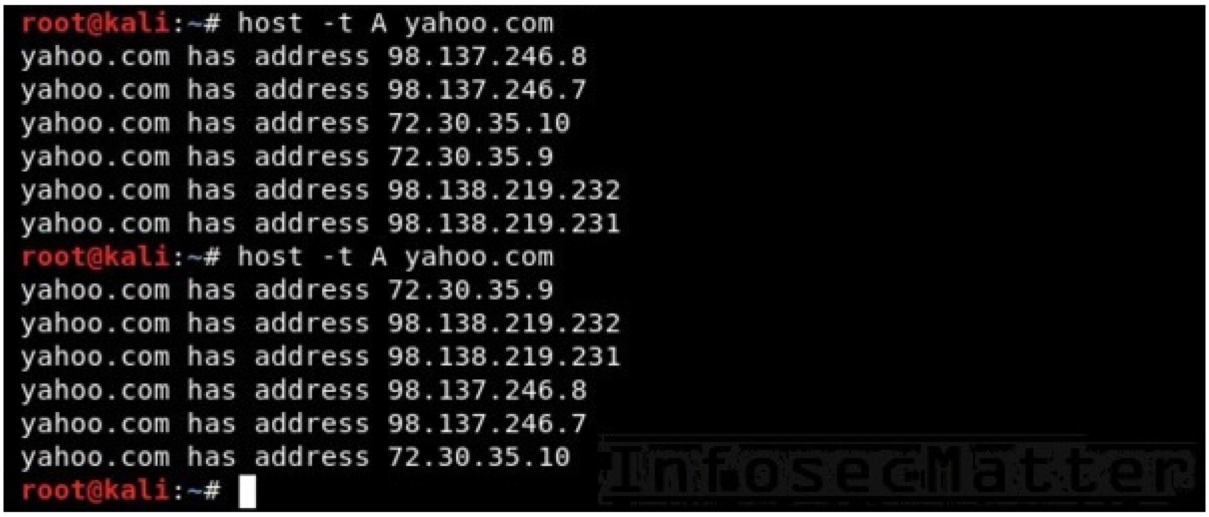
- Examining HTTP headers

- Examining SSL certificates
- Looking into historical DNS records
- Making the website send us an email
- Finding a vulnerability on the website
- Examining target's known network ranges and IP addresses

Detailed information on those techniques can be find on the following links:

Website on multiple IP addresses

Sometimes the target website is using multiple IP addresses and some form of load balancing. For instance, the following screenshot shows a website using load balancing using DNS round robin. Notice how the IP address order changes every time we do the DNS lookup:



```
root@kali:~# host -t A yahoo.com
yahoo.com has address 98.137.246.8
yahoo.com has address 98.137.246.7
yahoo.com has address 72.30.35.10
yahoo.com has address 72.30.35.9
yahoo.com has address 98.138.219.232
yahoo.com has address 98.138.219.231
root@kali:~# host -t A yahoo.com
yahoo.com has address 72.30.35.9
yahoo.com has address 98.138.219.232
yahoo.com has address 98.138.219.231
yahoo.com has address 98.137.246.8
yahoo.com has address 98.137.246.7
yahoo.com has address 72.30.35.10
root@kali:~#
```

In cases like this it is important to port scan all the IP addresses in order to achieve full coverage. Some of those systems may be configured differently than the others and by scanning all of them, we may discover something significant.

So at this point we should have collected all the IP addresses that belong to the target website. Now let's proceed to perform the actual port scanning.

Port scanning strategies

When it comes to port scanning, we may have to consider different strategies depending on the nature of the test.

White-box / VAPT scenario

During a **white-box** testing or in a typical VAPT (Vulnerability Assessment / Penetration Testing) scenario, the objective is generally to find as many vulnerabilities as possible.

Therefore, we always scan **all ports** on the remote hosts which ensures that the coverage is full and that we don't miss any services running on the target hosts.

The following command is the typical way of how we port scan a website during a VAPT exercise:

```
nmap -n -Pn -sS -p0-65535 -oA output <IP-ADDRESS>
```

- -n (no DNS resolving)
- -Pn (no host discovery, treat the host as online)
- -sS (perform TCP SYN scan)
- -p0-65535 (scan all ports)
- -oA (output in all 3 formats – nmap, gnmap, xml)

From the timing perspective, we keep it default which means that we use the “normal” timing template (-T3). This is definitely not covert and so it is very likely to be detected.

We do it like this only when we know that we are not going to get blocked. Typically we ask the client beforehand to **whitelist** our source IP address, so we don’t get blocked.

This is also beneficial for the client and their SOC (Security Operations Center) team to discern between our testing activities and a potential real attack activity.

Black-box / Red-team scenario

Now In more **black-box** approach or in a **red-team** exercise, we would proceed differently. The objective here is not anymore to find as many vulnerabilities as possible, but of course we would like that. We would still want to achieve full coverage.

In these scenarios we also have to be covert in order to not get blocked. If we get blocked, then our results will be imprecise and incomplete. We will also “burn” our source IP address and we may have to get a new one (e.g. rent a new VPS (Virtual Private Server) instance).

So in order for our port scan to be covert, the most important aspect is the speed. We simply have to do it slowly enough to stay under the radar. These are our options with Nmap:

Nmap option	Timing template name	Speed
-T2	polite	1 packet every 0.4 seconds
-T1	sneaky	1 packet every 15 seconds
-T0	paranoid	1 packet every 5 minutes

For more detailed information on the timing templates, please see Nmap’s official documentation (link).

So, if we want a full coverage port scan within a reasonable time frame, we have to use the “polite” template (-T2) at minimum. This could finish the job in 7.2 hours. The command would be:

```
nmap -T2 -n -Pn -sS -p0-65535 -oA output <IP-ADDRESS>
```

But this would probably still be detected! The alternative is using the “sneaky” template (-T1) which would take 11 days to do the job. Do you have 11 days to perform a full port scan? It is still doable, but if we don’t have that, then we have to compensate. We have to find a balance between coverage and required time.

Finding the right balance..

The following 5 examples are what we typically use in a black-box or red-team exercises. We generally use these commands when we are not being whitelisted by the client. Depending on the project and allocated time frame, we need to pick the most suitable scanning strategy.

The following commands use the “sneaky” timing template (-T1) to avoid detection but still to provide results in a reasonable time frame. Note that we are only scanning TCP ports.

Feel free to use any of these commands according to your current situation and needs:

Scan for web server ports (56 ports)

```
nmap -T1 -n -Pn -sS -p80-90,443,4443,8000-8009,8080-8099,8181,8443,9000,9090-9099
-oA output <IP-ADDRESS>
```

Required time: 14 minutes.

Scan for web server ports and other interesting ports (93 ports)

```
nmap -T1 -n -Pn -sS -p21-23,25,53,80-
90,111,139,389,443,445,873,1099,1433,1521,1723,2049,2100,2121,3299,3306,3389,3632,
4369,4443,5038,5060,5432,5555,5900-5902,5985,6000-6002,6379,6667,8000-8009,8080-
8099,8181,8443,9000,9090-9099,9200,27017 -oA output <IP-ADDRESS>
```

Required time: 24 minutes.

Scan for 200 most popular ports

```
nmap -T1 -n -Pn -sS --top-ports 200 -oA output <IP-ADDRESS>
```

Required time: 50 minutes.

Scan for 1000 most popular ports

```
nmap -T1 -n -Pn -sS --top-ports 1000 -oA output <IP-ADDRESS>
```

Required time: 4 hours and 10 minutes.

Scan for 4790 most popular ports (similar to Nessus scanner)

```
nmap -T1 -n -Pn -sS --top-ports 4790 -oA output <IP-ADDRESS>
```

Required time: 20 hours.

Tips for long-running port scans

Run them safely

If you are using a dedicated Linux VPS for your activities, make sure that you run the port scans in a terminal multiplexer such as GNU Screen or tmux.

This will prevent you accidentally interrupting your scans in case there is connection disruption between you and your VPS.

Detect blockage

During the port scanning, it is good practice to check once in a while whether your scans are not being detected and whether you are not being blocked.

To do this, we can simply check whether we can still connect to the web server port that we know is open (e.g. port tcp/80).

We can easily do this using the Netcat utility like this:

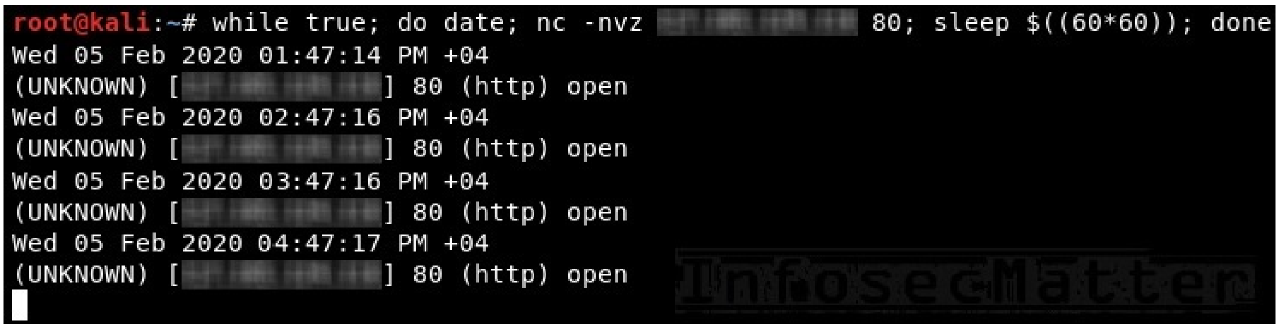
```
nc -nvz <IP-ADDRESS> 80
```

- -n (no DNS resolving)
- -v (verbose output)
- -z (zero-I/O mode, report connection status only)

If you would like to do this port check automatically say every hour, you could simply spawn another screen/tmux terminal window on your VPS and execute this one-liner in there:

```
while true; do date; nc -nvz <IP-ADDRESS> 80; sleep $((60*60)); done
```

Now you will have a good situational awareness report every hour:



```
root@kali:~# while true; do date; nc -nvz [redacted] 80; sleep $((60*60)); done
Wed 05 Feb 2020 01:47:14 PM +04
(UNKNOWN) [[redacted]] 80 (http) open
Wed 05 Feb 2020 02:47:16 PM +04
(UNKNOWN) [[redacted]] 80 (http) open
Wed 05 Feb 2020 03:47:16 PM +04
(UNKNOWN) [[redacted]] 80 (http) open
Wed 05 Feb 2020 04:47:17 PM +04
(UNKNOWN) [[redacted]] 80 (http) open
```

By the way..

Did you know how Nmap uses the '**–top-ports**' option and which those top (most popular) ports are? Easy! Simply run the following one-liner to find out the top 100 TCP ports:

```
sort -k3 -nr /usr/share/nmap/nmap-services | grep '/tcp' | head -100
```

It can be really helpful to know some of the shell scripting, don’t you agree?

Conclusion

We truly hope that this article provided all the necessary details on how to properly port scan a website. By following the 3 rules mentioned above, port scanning a website can be really easy and straightforward.

If there is something missing or anything unclear, please let us know in the comment section. Let us also know how do you port scan a website during your penetration tests.

Website