GaN Network Lab 3 (graded)

Introduction

Our goal with this third lab is to use powerful measurement tools that open the possibility to run large scale measurements.

Learning objectives:
- Discover, understand and use high speed probing algorithms
- Use ZDNS for CDNs infrastructure discovery
- Use ZMap to run scan of the IPv4 address space on specific port number

Content:
- Run simple measurements with ZDNS to better understand the different parameters
- Build a methodology to discover Google replicas
- Use ZMap

High speed probing introduction

As you may have understood, RIPE Atlas suffers from limitations in terms of probing rate and number of concurrent measurements that limit the scale of the measurement one can make with the platform. More specifically, we did measurements to get the result of the redirection using DNS and then traceroutes to obtain the path between a vantage point and the redirected server. But what if we wanted to get access to the result of the redirection for a client that is *not part of the RIPE Atlas platform?* Imagine that our goal is to find all the replicas of Google CDNs, what guarantee do we have that we discovered all its replicas after running DNS measurements from all vantage points?

Additionally, what if our goal was to discover all the paths from one point to all /24 prefixes (around 15M destination)? Or to discover all servers for a specific port for a service? This is not possible with RIPE Atlas. All these measurements could be useful for an operator to troubleshoot its path towards millions of destinations, for researchers to better understand the adoption of certain protocols, etc.

For these reasons (and many others), researchers have built powerful tools, running at high speed (hundreds of thousands of packets per second) to run *Internet scale measurements* (i.e. measurements for the entire IPv4 address space), starting with ZMap in 2013, followed by Yarrp in 2016 and ZDNS in 2022. Before the release of these tools, researchers and operators had to rely on network tools that you may have heard about, such as: Nmap, Traceroute and Dig. Because of their design, these tools are slow and require *days* to run Internet scale measurements from multiple vantage points.

Getting access to the lab environment

We cannot perform measurements from the machine of the university because of the firewall blocking most of UDP/ICMP traffic. Instead, connect to one of the three VMs with IP addresses 132.227.213.3<**J**> (J between 1 and 3). Connect with the command:

```
ssh user<I>@132.227.213.3<J>
```

Replace I and J by respectively the number of the user (1 to 10) and the number of the VM (1 to 3). Once connected to the VM, you will be able to execute commands **with root privilege, we kindly ask you NOT to perform unattended commands or services. If abnormal traffic/processes are detected, action will be taken…**

Note: You can also run these measurements from your local laptop but you will have to install the tools.

ZDNS: high-speed DNS probing

For this first section, we will study and use ZDNS, a high speed DNS tool. Without going into the detail, ZDNS leverages Go high parallelization to start thousands of routines (called go routines) to run DNS queries in parallel. Before using ZDNS, let's run some DNS queries using Dig, an historical tool:

```
dig www.google.com
```

- By default, what is the query class used by Dig?
- What type of DNS record is being queried? Do you know other types of DNS records? (give at least two and the service they direct to)
- What is the transport protocol used by DNS? Is it the only possibility?
- How many domain names can you resolve at once? What do you think of using Dig for getting the resolution of thousands of domain names?

Now run the same DNS query with ZDNS:

```
echo “www.google.com” | zdns A
```

- Did you notice any difference?

Now run the following command:

```
cat /srv/top-1k.csv | zdns A --threads 200
```

The file /srv/top-1k.csv contains the top one thousand most queried domain names. You can consult it with the command:

```
cat /srv/top-1k.csv
```

Note: The “nature” of these websites was not checked, we are not responsible for which websites people are connecting to…

- What is the purpose of the **threads** parameter?
- What do you think about the time ZDNS took to run the query? How many domain names were resolved per second?
- By default, which resolver is used by ZDNS? What kind of ethical concern can you think of when using such a tool? (think in terms of traffic load, cache size, etc…)
- On which IP address is the resolution made by the authoritative name server? What is the name of the entity making the request?

Now that we know ZDNS can execute high speed probing, let's dive into the details of the resolution and build a methodology for getting access to the resolution for *any IP addresses on the Internet*:

- Perform DNS resolution on www.google.com but this time using Google Public DNS resolver. Which parameter did you change?
- Do you obtain the same resolution when using ZDNS default resolver?
- Why, in some cases, is it problematic to use a large public resolver for making DNS queries?

Google public DNS resolver is addressed with an Anycast IP address making it difficult to know to which replica you were redirected to. Nonetheless, each of these replicas also have a unicast IP address that you can retrieve using the following command:

```
echo "o-o.myaddr.l.google.com" | zdns TXT --name-servers 8.8.8.8
```

- What is a TXT record?
- Where is located the IP address of the Google public DNS replicas you are redirected to? Can you change the redirected replica?

Instead of checking online, you can also check the location of all Google public DNS replicas and their unicast prefixes with the command:

```
echo "locations.publicdns.goog" | zdns TXT --name-servers 8.8.8.8
```

- What do you think about the location of the Google public DNS replica you are being redirected to?

For now, we used a different resolver but this does not mean that we can get access to the result of the resolution for another IP address than the one that we own. Run the following two commands:

```
echo "www.google.com" | zdns A --name-servers 8.8.8.8 --client-subnet
23.148.232.0/24
```

```
echo "www.google.com" | zdns A --name-servers 8.8.8.8 --client-subnet
192.134.0.0/24
```

- Do you recognize these IP prefixes ? If not, check their location online.
- Do you obtain the same resolution? Where are located the returned IP addresses?

Still using Google public DNS, run the DNS resolution for the two previous prefixes but this time for reuters.com:

- Do you obtain different responses for the two prefixes?
- What is the value of the **source_scope** parameter?
- Using ZDNS documentation or searching online, what is this **client-subnet** parameter? What is it used for?
- How does this parameter influence the result of the redirection? Why is it being used by CDNs?
- What is the **source_scope** parameter in the response?
- Perform the same query but this time using Cloudflare public DNS resolver (you can also try Quad9 if you have time)
- What can you conclude about the support of the **client-subnet** parameter by Cloudflare?
- Is it coherent with the way Cloudflare redirect its client to one of its replicas?
- Why is Google supporting it?

- Based on this **client-subnet** parameter, propose a methodology to enumerate (i.e. discover all replicas) the CDN of Google.

- How many of the top one thousand most queried websites are supporting this parameter?
    - You can save the output of the resolution into a file using the **-o <dir-output-file>.json**

    - You can also retrieve the output file on your local machine using the **scp** command and then parse the results.

```
scp user<I>@132.227.123.3<J> <path-on-your-local-computer>
```

ZMap: high-speed network scanning (not graded, for fun)

ZMap is a powerful tool to run exhaustive IPv4 scans (4 billion IP addresses) from a single machine in under an hour (if your network is sufficiently provided). It can send up to millions of packets per second based on three main principles:

- **Statelessness:** Zmap does not keep in memory the state of each probe sent during the measurement (where it was sent, since when, etc.). This makes ZMap fast because of very low processing and memory usage (as opposed to historical tools such as Nmap)
- **ZMap** uses **deterministic pseudorandom permutations** to generate probe addresses on the fly and relies on matching replies by metadata (like source port, sequence number, etc.), not per-host records. This principle allows to filter received traffic not triggered by the measurement and to probe only once (there is no retry) every scheduled IP address.

This allows ZMap to send millions of packets per second with very little memory.

Run a network scan with ZMap using the following command **DO NOT SIMPLY REMOVE OR CHANGE THE -r PARAMETER. BY DEFAULT ZMAP USES ALL AVAILABLE BANDWIDTH AND YOU WILL BREAK THE VM OR WORST**:

```
sudo zmap -p 23 -r 1000 --max-results=50
```

- What is the application running on port 23? port 22?
- What kind of security risks are linked with this application?
- What is the **max-results** parameter?
- What kind of ethical concerns arise when using ZMap?
- How does ZMap avoid being considered as a DDoS attack by the probed destination ? (We are probing at thousands of packets per second).
- How does the traffic look like from the standpoint of the operator hosting the vantage point running ZMap?

Bibliography:

Papers:

- ZDNS
- ZMap
- Yarrp

CDNs DNS:

- Google public DNS
- Cloudflare public DNS

Others:

- IPInfo
- PeeringDB