Implementation of a network management sysytem

Hugo Guimarães
Faculty of Engineering of
University of Porto
Porto
Email: up201806490@edu.fe.up.pt

Ricardo Nunes
Faculty of Engineering of
University of Porto
Porto
Email: up202109480@edu.fe.up.pt

Pedro Ponte
Faculty of Engineering of
University of Porto
Porto
Email: up201809694@edu.fe.up.pt

Xavier Pisco
Faculty of Engineering of
University of Porto
Porto
Email: up201806134@edu.fe.up.pt

***Abstract*—This paper was written in order to explain the implementation of a network management system similar to an organization, like FEUP. Based on an initial network diagram, we started the implementation, trying to automate the processes as much as possible, so it can be easy to start or restart if an error occurs.**

I. Introduction

The main goal of this project is to implement a network management system similar to an organization. This network should allow remote users to access the organization's servers, and its internal servers to connect to the world wide web through a proxy server, as well as provide access to its public services on the internet (for example, webmail and web server).

As a starting point for this project, we used part of the work carried out during Lab 3, where we had to implement a network of an organization with services similar to those we wanted in our own.

II. Related work

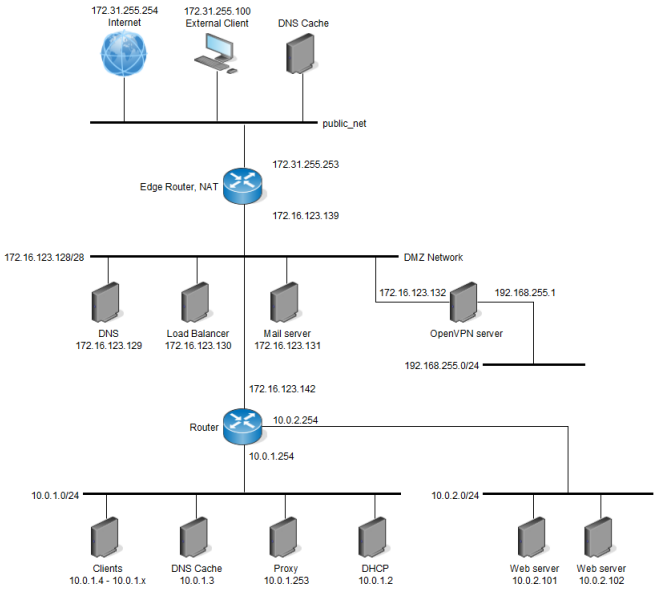
Organization networks have been implemented for a long time and each organization has its own approach with its own use case modifications, which brings some differences in implementation, but with similarities in goals. The closest example we found is the *Cisco* campus solution design guide [1]. There, they designed a system that centrally manages major workflow areas, which includes network site profiles for physical devices, domain name system (DNS), Dynamic Host Configuration Protocol (DHCP), IP addressing, software image management, plug-and-play configurations, and user access. They also provisioned devices for the management and creation of fabric domains, control plane nodes, border nodes, edge nodes, fabric wireless, local-mode wireless, and external connectivity. Throughout the article, we see that there is a big focus on automation of infrastructure provisioning and instancing, which are common goals of our own solution.

III. Networks

At the core of our solution, we had to create all the needed networks for our project to be similar to the existing network of the FEUP's Lab. In these labs, it is needed to have a network for the students and teachers that bring their personal computers, which we called the *clients network*, another IP range for all the computers and servers that are inside the lab, the *servers network*, and, at last, a network for the services that need to be accessible to everyone inside or outside the Lab, the *DMZ network*.

A. Network Diagram

Our solution can be graphically represented through the following diagram.



B. Clients Network

In the *clients network*, we have all the personal computers and smartphones that people will bring to the lab. These

devices are connecting and disconnecting to the Internet as they come in and leave the lab so that has to be taken into account. This network is implemented for 250 different clients.

It has five different types of machines, the *clients* which have different IPs for each one of them, a *DHCP* server with the IP 10.0.1.2, a *Proxy Server* with IP 10.0.1.253, and *DNS Cache* at 10.0.1.3 and *access to a router* at 10.0.1.254.

1) DHCP: Since the computers in this network are constantly changing, it's important to dynamically assign an IP address to the clients that are entering the network. For that, we used the DHCP protocol. This step is crucial to allow communication in our network through the IP protocol as it ensures that all the clients that connect to the network have a unique IP and the needed routes for internet access.

2) Proxy: The proxy server is an intermediate between our organization's clients and the servers that provide the resources requested to the network. That allows us to evaluate the clients' requests and perform the required transactions through the proxy, providing us a way to accept requests only in the local network, which in turn prevents direct HTTP traffic from clients to somewhere outside of the client network, which in this specific case is 10.0.1.0/24.

3) DNS Cache: The DNS cache is a Bind 9 server that simply forwards all the DNS resolve queries to the DNS server, which is implemented in the DMZ network. If needed, it may store private domain names that can only be accessed by the users inside this network, but, for now, it redirects the queries in order to have the same DNS access as computers from outside the Lab.

4) Router: The router is the gateway between this network and the other networks in the lab. It is connected to all the three previously mentioned networks and implements NAT for this network to enable replies from other machines outside the network.

C. Servers Network

The servers' network is a simple network that needs to be accessible for everyone that is in the Lab but must not be accessible to computers that are outside the network unless they are connected via VPN.

This network was created to have 254 possible connections, and so we gave it the IP range 10.0.2.0/24. Included in these IPs we have a router that has IP 10.0.2.254, this enables the servers inside the network to have access to the internet so they can, for example, be updated regularly.

In this network, we have 2 example servers that are running a simple Nginx web server for test purposes.

1) Web Server: Our web server is present to represent the organization's website or web application that is self-hosted and should be open to public access through our Load Balancer, which is present in the DMZ Network. Due to the nature of our project, this is nothing but a mock-up application, however, it is configured and its docker container is running.

D. DMZ Network

This network is a network that can be easily accessed by both the internal networks, the clients and the servers, and the internet.

It is a logical sub-network that is used to connect hosts that provide an interface to an untrusted external network. It exposes an organization's external-facing systems, separating them from the private sub-network that must be secured.

This is the network that has all the servers that need to be accessed by the computers outside the lab, so it currently has the Mail server (172.16.123.131), Load Balancer for the web servers (172.16.123.130), a VPN server (172.16.123.132) and a DNS for those servers (172.16.123.129).

1) Load Balancer: This acts as a reverse proxy while at the same time distributing the incoming traffic across several servers that serve the same purpose. This was implemented to provide a way for our organization network to increase its capacity to deal with concurrent users as well as increase the reliability of its applications. The only application that currently benefits from this is the Web Server.

2) DNS: The DNS is present in our organization's network to provide a way for our browsers to get to internet resources by using their domain names. While providing the domain names to our requests, the DNS system will translate them into their respective public IP addresses.

3) Mail server: Our webmail is present to represent the organization's self-hosted electronic mailing services and should be open to public access, therefore it is present in our DMZ network. Due to the nature of our project, this is nothing but a mock-up application, however, it is configured and its docker container is running.

4) VPN: Our organization must allow clients to connect to the network and access their resources, easily and securely, even if they are not hosted locally. According to a "perimeter 81" online publication [2] and sustained by the lectures in classes, the best way to provide this is with a VPN service.

Therefore, we created a VPN network in the DMZ sub-network (172.16.123.139), which authenticates clients using a password for a Certificate Authority and a private key.

To build this VPN, a docker OpenVPN container was created, which is an open-source robust, and transparent security protocol that allows connecting multiple clients for testing purposes.

5) NAT: As it's impossible to have a unique public IP address for each computer to connect to the Internet, we have to implement NAT (Network Address Translation) in our router.

Basically, NAT is a method of mapping an IP address space into another by modifying network address information in the IP header of packets while they are in transit across a traffic routing device. This way, each organization has a public IP, and all devices have access to the Internet using that public address. When a device on the local network wants to send a packet, it delivers to the router, and the NAT will take care of changing the packet's local IP for the public IP. In return, the

same thing happens, but now the public IP will be replaced by the local IP of the device that made the request.

6) Edge Router: That is our organization's specialized router which is located at the boundaries of our system, which is the server's network. This router's function is to provide a way to enable our network to establish communications with external networks.

IV. Project organization

In order to have a changeable and fixable network, we tried our best to automatize the whole process and make it as simple to execute as possible.

Our final result was a group of bash scripts separated by files in an organized folder structure that can be found at https://github.com/Hugomguima/GR-Proj. We had two main goals when organizing the files: automation and modularity.

A. Automation

After restarting the computer and setting up all the bridges, one for each of the physical networks, we simply need to execute the "run.sh" file in one of our personal machines while connected to the computer and it will automatically copy and execute the needed files in the corresponding virtual machines. After that, the only non-automatic part is creating the passwords for the VPN server.

This is possible by having scripts that copy the files via ssh and have the scripts call each other in the needed order.

B. Modularity

To enable extensibility in the network we decided to separate the process of starting a machine in the network into three different parts.

Firstly, we have the build phase in which, if needed, the user should create a folder inside the "dockers" folder where it has a "build.sh" file which will build the docker image needed for the container.

After that, the user should create a launcher in the "launchers" folder, simply create a script that will launch the machine in the network.

At last, you need to choose which network it will be on and, on the file corresponding to that network add a call to the launcher script that was created before.

C. Update and/or Reboot

When one of the machines has to be rebooted there are two options, either a simple reboot or a reboot which is also an update.

If one of the machines just needs to be rebooted you simply need to kill the docker container that is running and execute the "launcher" script associated with that machine in the virtual machine you want, either B or C.

If you need to update or change configuration files it may be needed to run the "build.sh" script that corresponds to the docker image of the machine before doing the same steps as a normal reboot.

When stopping machines and starting again some cautions may be needed, especially the order in which the machines are turned on again. As an example, if you want to reboot the DHCP server and create a client you should first fully restart the DHCP server and then add the client, if the client is added in the middle of the DHCP reboot, the client probably won't have a correct IP address, thus it won't have access to any network.

V. Evaluation

When the script ends and everything is set and done we needed to know if everything was running as it should or if there was any type of problem.

Firstly, we decided to check communication from the computers inside the private networks to every other network it should reach. We decided to check if the client was able to ping all the hosts he was supposed to and if it could use DNS in order to ping those devices.

Therefore, we created a script called "connection.sh", which attempted to ping to a provided IP.

We also created the function *assertTrue* and *assertFalse*, which would crash the program in case the ping didn't work or worked, respectively.

A. Client

The following script displays all the connections that the client should be able to have.

```
./cnct.sh client1 10.0.1.254; assertTrue
./cnct.sh client1 10.0.2.101; assertTrue
./cnct.sh client1 172.16.123.129; assertTrue
./cnct.sh client1 172.16.123.130; assertTrue
./cnct.sh client1 172.16.123.131; assertTrue
./cnct.sh client1 172.16.123.132; assertTrue
./cnct.sh client1 172.16.123.139; assertTrue
./cnct.sh client1 mail.myorg.net; assertTrue
./cnct.sh client1 1.1.1.1; assertTrue
./cnct.sh client1 google.com; assertTrue
```

B. Load Balancer

Secondly, we chose to check if the machines on the DMZ network had access to both the machines in the private networks and the outside world, so we tested if the "loadBalancer" had the following connections:

```
./cnct.sh loadBalancer 10.0.1.3; assertTrue
./cnct.sh loadBalancer 10.0.2.101; assertTrue
./cnct.sh loadBalancer 1.1.1.1; assertTrue
./cnct.sh loadBalancer google.com; assertTrue
```

C. External Host

Without further authentication, attempting to connect to our network should allow access to the public section of our network, but should refuse all connections attempted to private network components.

Therefore, we chose to check if an external host could successfully connect to the DMZ and the mail service. In contrast, it must fail when communicating with an internal client and the web server.

```
./cnct.sh ext_host 10.0.1.3; assertFalse
```

```
./cnct.sh ext_host 10.0.2.101; assertFalse
./cnct.sh ext_host 172.16.123.132; assertTrue
./cnct.sh ext_host mail.myorg.net; assertTrue
```

D. External Host With VPN

Finally, we connected the external host to the VPN and tried the exact same connections as before. This time, it should be able to connect with not only the machines in the DMZ but also the machines in the private networks.

```
./cnct.sh ext_host 10.0.1.3; assertTrue
./cnct.sh ext_host 10.0.2.101; assertTrue
./cnct.sh ext_host 172.16.123.132; assertTrue
./cnct.sh ext_host mail.myorg.net; assertTrue
```

These tests are being executed immediately after everything is up and running but they can be executed at any time to check if the system is still up or if anything has stopped working.

VI. Conclusion and Future Work

A. Conclusion

We were able to reach our main goal of building a Network similar to an organization, which presented itself as an enriching experience that improved our knowledge about several components of a large network, such as the DNS and the VPN, and it also made us realize how each of them is connected with one another in harmony in order to create a fully automated working system, which made us realize how crucial it is for an organization to have a structured network.

Although the network was successfully created, we faced several difficulties, mostly associated with the lack of detailed online information that would allow us to directly solve our problems.

B. Future Work

Even though we are pleased with the final state of the project, we realized that there are several features whose further implementation would improve the quality of our network, such as:

- Firewall - The addition of a firewall would allow the company to filter incoming and outgoing traffic.
- Security Analysis - To ensure security's best practices are being followed.
- Wireless Intrusion Prevention System (wIPS) - To monitor the presence of unauthorized access points.
- Infrastructure Monitoring - To capture the use of each device

References

[1] "Campus LAN and Wireless LAN Solution Design Guide." [Online]. Available: https://www.cisco.com/c/en/us/td/docs/solutions/CVD/Campus/cisco-campus-lan-wlan-design-guide.html

[2] "What is an Enterprise VPN Solution ?" [Online]. Available: https://www.perimeter81.com/resources/enterprise-vpn-solution