* In this homelab I mplemented and configured DHCP (including multihomed), DNS, and DDNS services as part of a network administration project. Applied industry best practices to ensure secure and reliable service delivery.
* DHCP
* DHCP multihomed
* Security best practices for DHCP
* DNS
* DDNS
* Best security practices for DNS
* NATING
* Topology :

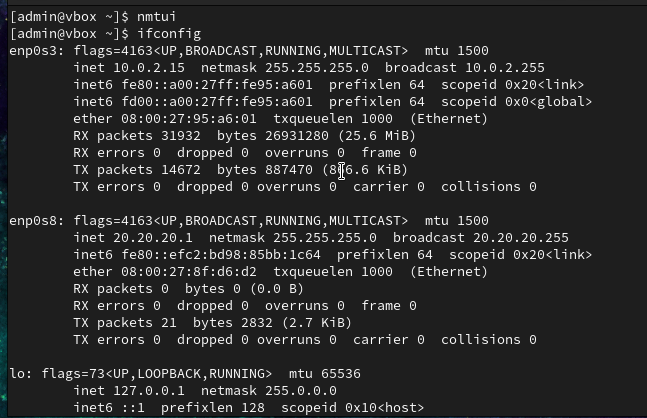
\_\_ centos9 VM with internal network interface / NAT interface for internet connection, and it will be the server

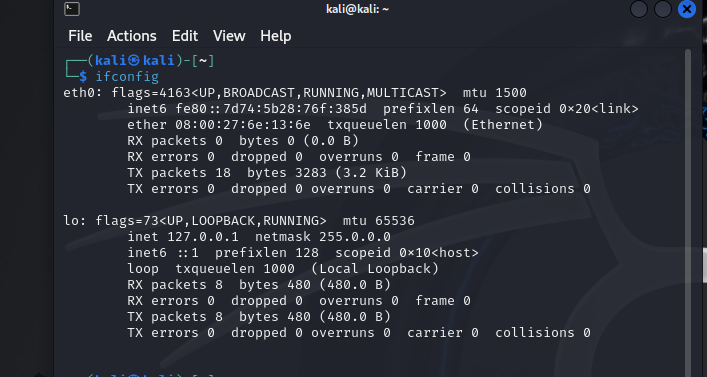
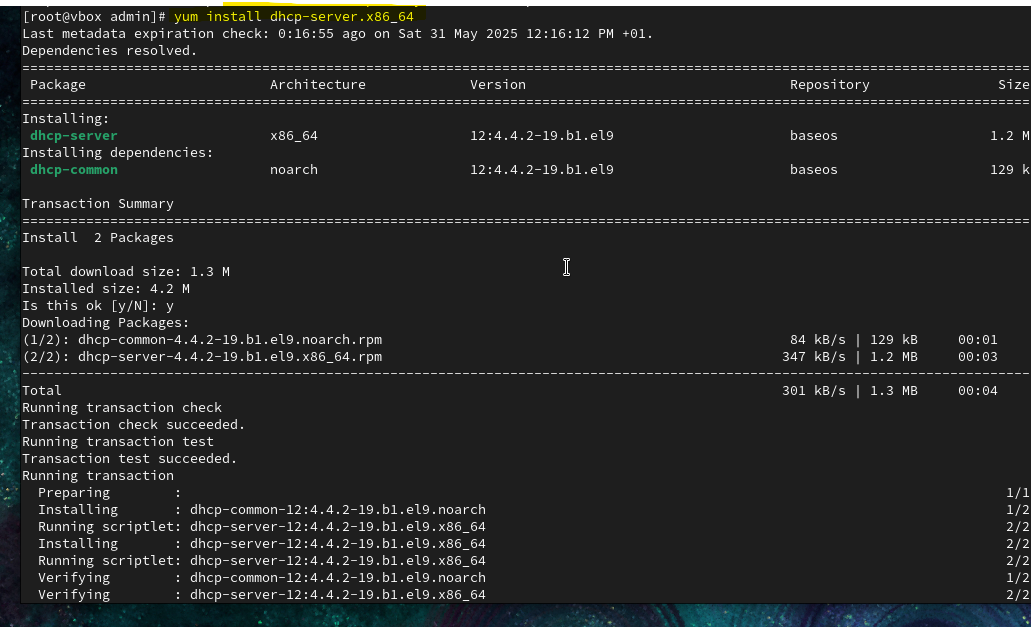
\_\_ kali VM with internal network interface same as the centos VM so they can be in the same LAN, and it will be the client

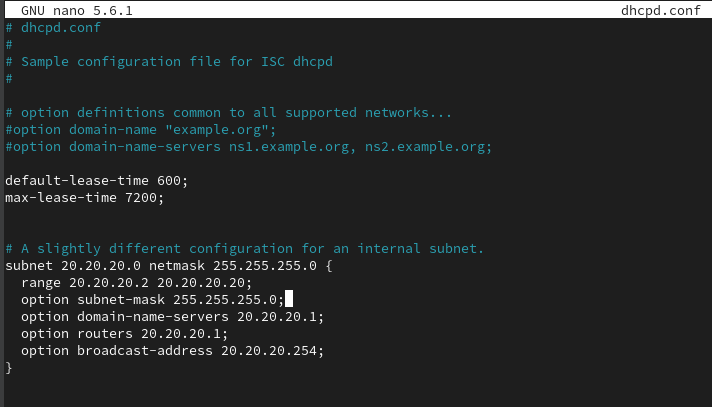
* DHCP
* Set a static ip to the internal interface on centos, so later, this interface will be from where the DHCP will listen on and response from
* I use nmtui for that



* Now we have enp0s8 with ip 20.20.20.1/24 and 20.20.20.0/24 will be the subnet of the internal network



* And for our linux “client machine”, it has no ip and it should get a one after we configure the DHCP
* 
* Now lets update our system by “sudo yum install” and then lets install “dhcp-server” software for configuring the dhcp
* 
* Dhcp-server files architect: we do have the main file of our configuration “/etc/dhcp/dhcpd.conf”
* And the sample where we will refrence the configuration from “/usr/share/doc/dhcp-server/dhcpd.conf.example”
* Lets set a backup file of the dhcpd.conf and copie the sample file to the /etc/dhcp/ folder and start our configuration
* 
* Open the /etc/dhcpd.conf with your fav txt editor and lets take of all the sample options we don’t need and lets the ones we need to change them a bit to match our scenario
* We will keep the subnet declaration “we refer to it by block sometimes” and change only its options/parameters
* We will keep the lease time option out of the subnet declaration /block so it will be a global option “means it will be applied to every other subnet declarations/blocks if they exist”



* Subnet + netmask: Defines the network and subnet mask for this block
* Range : Defines the range of Ips “pool” that will be given to the clients “note that we exclude 20.20.20.1 by starting from 20.20.20.2 because its taken by the DHCP”
* Option router: default getway for the client, I seted it as the DHCP server IP because it will be the default getway after I enable NATING on it
* Option subnet-mask: declare the subnet mask, “optional, while we already declare it on (netmask )”
* Option broadcast-address is clear that its for broadcast address
* default-lease-time : Time (in seconds) for default lease duration
* max-lease-time : Maximum lease time that the server will allow
* tips:
* if we had a machine that we need to assign a static ip for it like a printer, create a “host declaration” and set as parameters the machines mac address and the ip you want to give to it:

host “chosen name” {

hardware ethernet 00:11:22:33:44:55;

fixed-address 192.168.1.50;

option host-name "my-server";

}

NOTE: sYou can set any other options u want for that block

* If we have other interfaces with a static IP on our DHCP server and from that other interfaces its linked with other subnets, and we want the DHCP server to serve the clients on that subnets we can just add another blocks/subnets declaration, just make sure to add those interfaces to works for the DHCP service

Dhcp first-interface other-interface

Or use DHCPDARGS="first-interface other-interface" in /etc/sysconfig/dhcpd

NOTE: while im writing this write-up I found that declaring the interfaces where the DHCP should wor on isn’t something necessary anymore because the dhsp-server does recognize that automatically now

!!! and that’s what multihomed DHCP is

* If we have one interface on the DHCP server and it linked with more than one subnet

So all the DHCP requests and Rrponses will be on that interface, and we want the DHCP to serves each subnet from that one interface, we should use “shared-network” feature, we mostly been set a relay-agent on the router or the firewall and that’s how it should be.

“A shared-network block groups **multiple subnets** that exist on the **same physical network interface or broadcast domain**.”

It will looks like this in the dhcps.conf:

shared-network “chosen name” {

subnet 20.20.20.0 netmask 255.255.255.0 {

range 20.20.20.2 20.20.20.20;

option routers 20.20.20.1;

}

subnet 192.168.2.0 netmask 255.255.255.0 {

range 192.168.2.100 192.168.2.150;

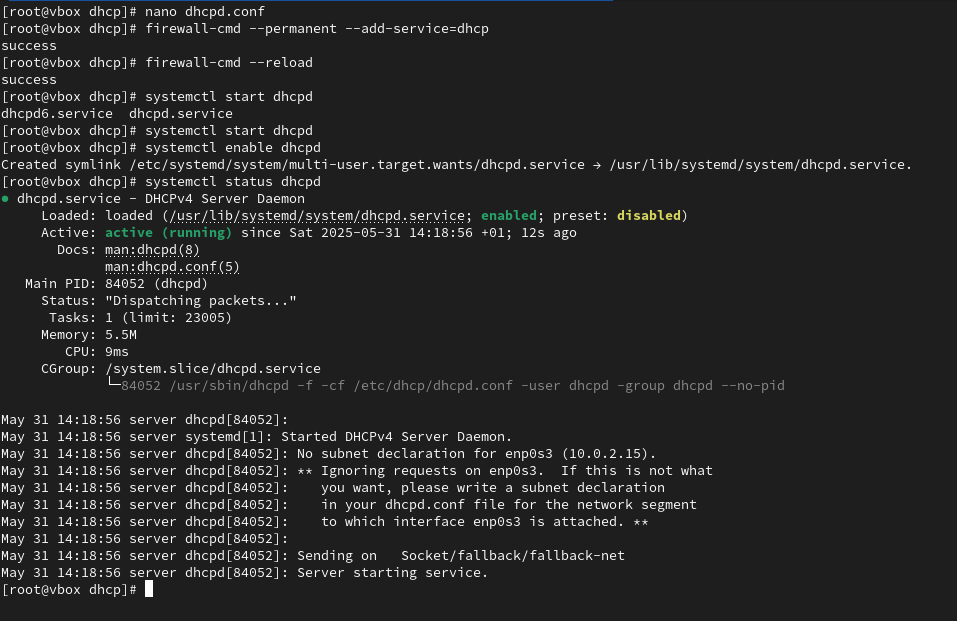
option routers 192.168.2.1;

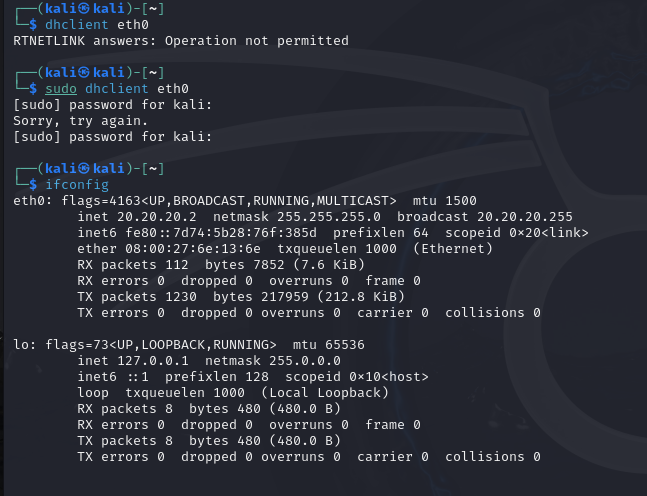
}

}

All the blocks are inside shared-network declaration

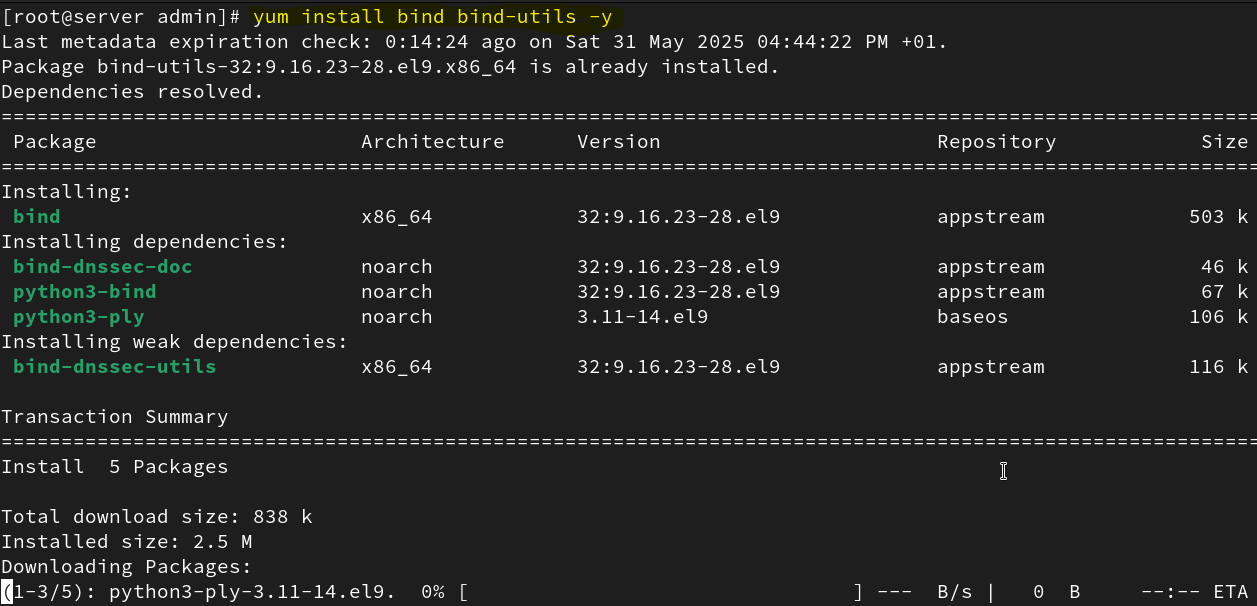
* Now lets back to our lab scenario, lets save the dhcpd.conf and enable the firewall for the dhcp-server and restart the service

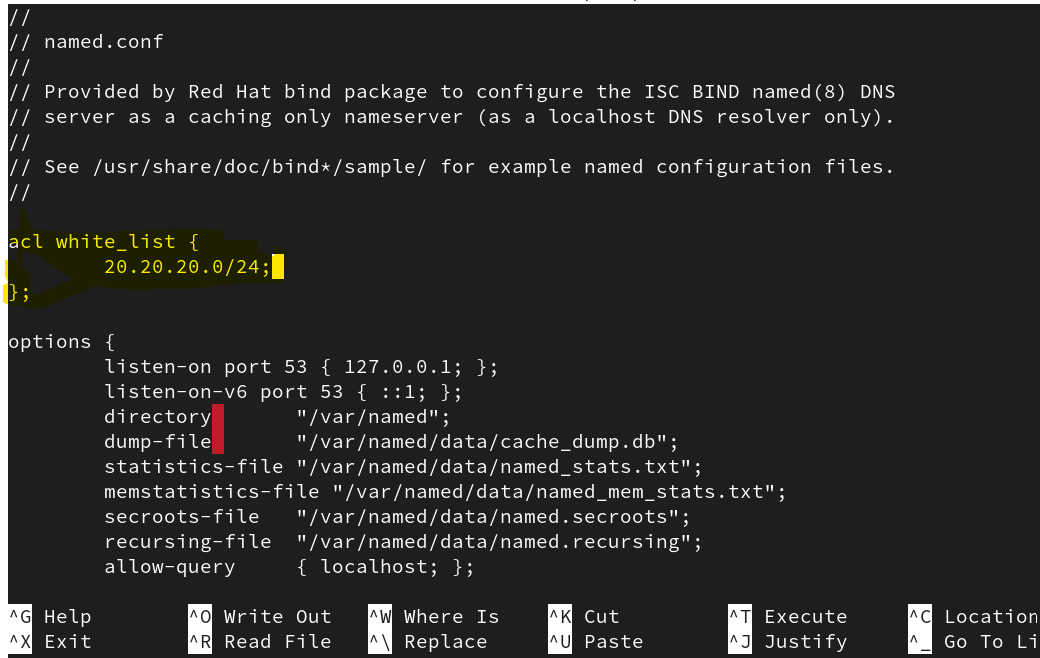


* Now the dhcp service is working
* Lets re-ask about an IP forour kali machin using “dhclient interface-name” 
* Now the dhcp service is totally working
* DHCP best practice “security” Tips:
* Better to secure the dhcp service from the DHCP starvation and DHCP spoofing attacks, using portsecurity mecahnismes on top of switches’ ports in the network, “dhcp snooping/dhcp trust port/dhcpsnooping limit….etc”
* Never set a dhcp service on the same server where you are setting the domain controller because:

1. If the DHCP serve guest user through the wifi networks, that’s bad because we are sure that we don’t want any kind of links between the guest users and our DC
2. Putting both on the same machine can lead to **performance bottlenecks** and makes **troubleshooting harder** when one service fails.
3. In a secure environment sys team manages DC and network team manages DHCP, so its better to separate them so we don’t **blurs responsibility boundaries**

* In big large networks its better to use a DHCP failover, hots-standby mode, or loadbalance mode
* Use DHCP reservation instead of setting a static IP to each static device on the network
* Exclude some Ips for future static devices
* DNS:
* On the same topology we will build DNS service, using bind9 software
* Bind9 files architecture:
* /etc/named.conf => maine configuration file
* /var/named => zone files, “zone files include dns records and how option that defines how the named deamon should deal with those records”
* /etc/endc.conf => the settings file for the rndc tool “it’s a tool we use to communicate with the named demon ” and its use a symmetric key for authentication
* /etc/rndc.key => the file where the rndc key exits in, change its read permission to root so only root can use it to communicate with named deamon
* Bind9 general work flow:
* In the /etc/named.conf we set our main config like: ACL statements, options statements “where we set parameters that control the dns behaves and its characteristics”, zone: the zone statement allows you to define the characteristics of a zone, such as the location of its configuration file and zone-specific options, and can be used to override the global options statements / revers zone same as zone but for resolving domains to IPs, include statement “we use it to import files instead of writing its content inside the named.conf so we can separate files permissions”, view statements “we use when we want to separate how the dns should serve different networks/subents, imagine you have to diffrents subnets each one of them has different domain name, zone and they are should be treated differently and with different options and zones ”
* Also we have zones, where we set our dns records “domain names and IPs”
* Now lets start by installing the bind9



* Lets open the /etc/named.conf where most of our configuration will be “I use nano”
* lets ignore the sample for now , and lets create an access list “white list” where we define the subnet that we want to let him query the our dns server
* 
* NOTE: the access list can take : none, localenet, localhost, Ips subnets, any, as parameters
* Now lets jump to the options statements
* 
* Explain the non-obvious options rules:
* Directory: Specifies a working directory for the named service. The default option is /var/named/. , (e.g it where the named service will look for the zone file that we will declare later)
* Dump-file: is where the dns server caches the just resolved dns query, its useful for debagging and monitoring
* Statistic-files: its where the dns store how many queries it resolved…, useful for performance monitoring
* Memstatistics-file: stores the statistics of memory usage by the dns service
* Secroots-file: file that store the dnssec anchor key that verify the integrity of authoritative responses data
* Allow-query: list of which clients can query the dns server for authoritative records, it allows any by default, if u can see we set our ACL on it, and our localhost
* Allow-query-cach: list of which clients can query the dns server for recursive records, it allows localhost and localnet by default if we didn’t set it up, if u can see we set our ACL on it, and our localhost
* NOTE: To prevent distributed denial of service (DDoS) attacks, it is recommended that you use the allow-query-cache option to restrict recursive DNS services for a particular subset of clients only.
* Recursive: set it to yes if you want your dns to be a recursive one and to no if you want it to be authoritative one
* Forwards: who should we forward the recursive queries to
* NOTE: for more options visit <https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/7/html/networking_guide/sec-BIND#table-bind-namedconf-common-options>
* Now lets dig into the zone option, The zone statement allows you to define the characteristics of a zone, such as the location of its configuration file and zone-specific options, and can be used to override the global options statements. It takes the following form:
* zone zone-name {
* option;
* ...

};

The *zone-name* attribute is particularly important, as it is the default value assigned for the $ORIGIN directive used within the corresponding zone file located in the /var/named/ directory. The named daemon appends the name of the zone to any non-fully qualified domain name listed in the zone file. For example, if a zone statement defines the namespace for example.com, use example.com as the *zone-name* so that it is placed at the end of host names within the example.com zone file.

* For what options it takes look them up at <https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/7/html/networking_guide/sec-BIND#sec-bind-zone>
* Lets build our zone statement but, firstly I would like to take off the root zone from that file because I will let recursing queries to be handled after the dns forwards them to the 8.8.8.8 or 1.1.1.1 dns as we set up.

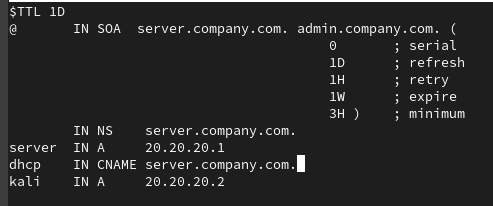


* I called the domain by company.com
* Now we should create the /var/named/company.zone file to set our records on it

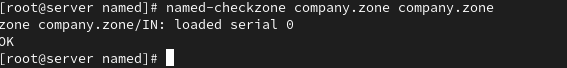
NOTE: we can verify if we done any mistake “like syntax mistake…” on this file by this command : named-checkconf “path to the named.conf”. if nothing appears after the command that mean all is good

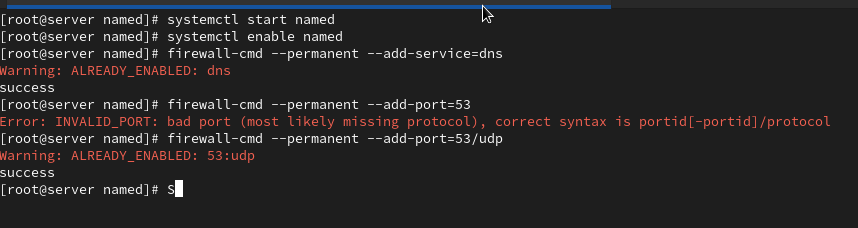


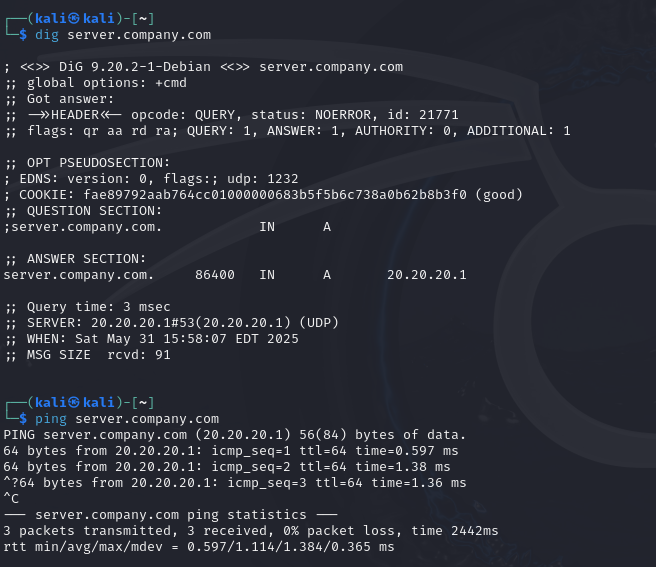
* Now lets jump into the /var/named and create company.zone and get its config from a sample and the we change on it but first here is a breakdown of what zone file mostly contain
* **$TTL**  
  This sets the default time (in seconds) that DNS records can be cached. For example, $TTL 86400 means records are valid in cache for 24 hours.
* **@**  
  The @ symbol is a placeholder for the current domain name defined in the zone. It usually refers to the main domain (like example.com).
* **SOA (Start of Authority)**  
  This is the first required record in a zone file. It defines the primary DNS server for the domain and administrative info. It includes:
  + The main DNS server name
  + The admin's email (written with a dot instead of @)
  + A serial number (updated whenever you change the zone)
  + Timing values (how often secondary servers check for updates)
* and then we set the records include the records type like:
* **NS (Name Server)**  
  Lists the DNS servers for the domain. These are responsible for answering queries for that domain.
* **A Record**  
  Maps a domain or subdomain to an IPv4 address. For example, www IN A 192.168.1.10 means www.example.com points to that IP.
* **AAAA Record**  
  Same as an A record, but for IPv6 addresses.
* **CNAME (Canonical Name)**  
  Creates an alias for another hostname. For example, mail IN CNAME www means mail.example.com points to www.example.com.
* **MX (Mail Exchange)**  
  Specifies the mail server for the domain. Includes a priority number to determine which server is preferred if there are multiple.
* **PTR (Pointer Record)**  
  Used for reverse DNS lookups. It maps an IP address to a hostname. Usually found in reverse zone files.
* **TXT (Text Record)**  
  Stores text data for a domain. Commonly used for SPF records or domain verification (e.g., for email services or Google verification).
* **Now lets build our zone file:**



* Save it and change the its owner to named with read and write as user and read only ad group, both for named, then lets verify if we have done any syntax mistake by the command:
* Named-checkzone “domain name” “zone file of the domain name”



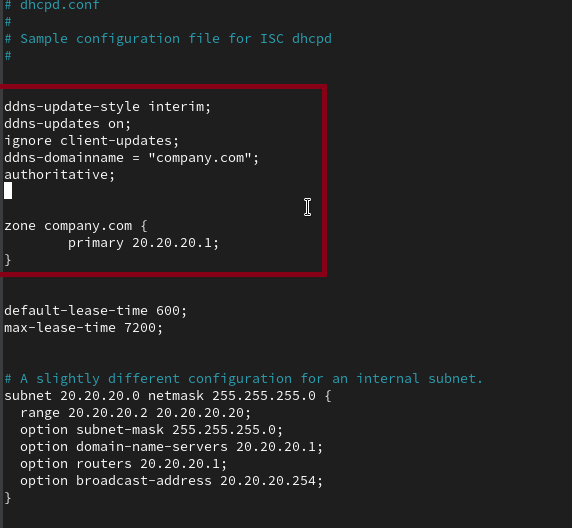
* OK message, means all good.
* Before running the service, by systemctl, one fault I made which is I didn’t create the company.com under /var/named , Iso I had to move it to there
* Now start and enable the named service, and enable it on the firewall, I already done that and I do it again so the firewall is telling me that I already done it, so don’t mind the red things. 
* Now from our kali machine, we know that we set up a dns to look through it while we were setting the DHCP config, now lets try to dig a domain name from our kali and see if the dns server we just set uo will resolve it



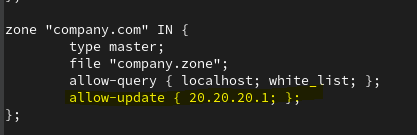
* As we can see it works , and we can ping the server.company.com which is the dns/dhcp server

NOTE: dig is a good tool for testing your dns server, see more on it here <https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/7/html/networking_guide/sec-bind#sec-bind-dig>

* Now imagine if we were in a large network, we wont set each individual device a record on its file zone, while what we want to do is that the dhcp should be the one who updates the zone records, and this called DDNS, while we dynamically update the dns records somehow, and in our case it based on the DHCP
* Now lets got back to the /etc/dhcp/dhcp.conf and add some parameters globally that describe the DDNS mechanism from the DHCP perspective



* Explain the parameters:
  + Ddns-update-style: the style or the way the dhcp will updates the zone records, interim is the best practice type, while we have none, and ad-hoc which is legacy
  + Ddns-update on: enable the updating of the dhcp to the dns zone records
  + Ignore client-adress: some clients when requesting an ip they send a domain name with it so the dhcp add them to domain zone, but in our case we want to ignore that
  + Ddns-domainname: describe the suffixes that will be add to the client’s name so then we store it on the dns zone
  + Authoritative: teels the dhcp that he is the only dhcp server that should update the dns zone records, so he want think that may be there is another dhcp that to depend on, in this mission
  + Zone “company.com” primary….: tells the dhcp who is the primary server of that zone
* Now save the file and lets allow the dhcp to update the record on that zone by adding a parameter in the zone statements in the /etc/named.conf



* This tells the dns to allow the one with “20.20.20.1” ip to update on the records of that zone
* SECURITY TIP: if the dhcp server is an internal server in the same network its ok to mention it by its ip like we just done, but if it were an external dhcp server, u would like to set a key for that, and to enable that based on who has the key, not just the ip
* Lets restart the dhcp and dns services

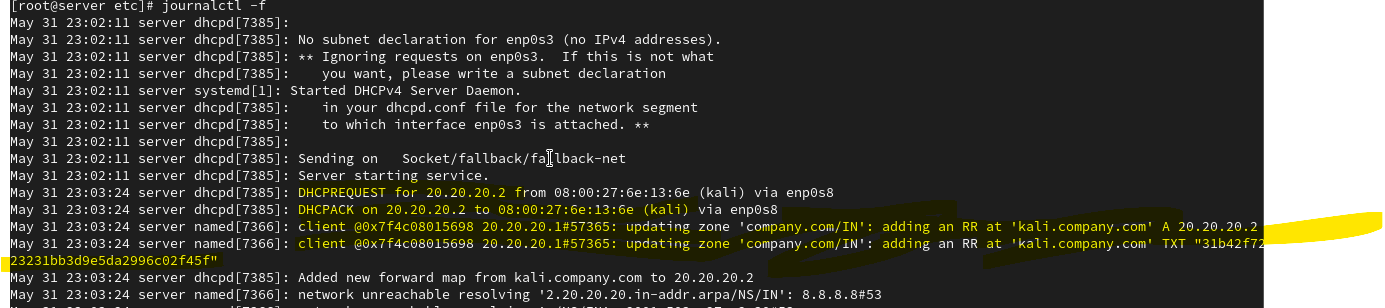


* So far so good, now lets try to reboot our kali and see if it can get and ip and see if it will be pushed to the zone records

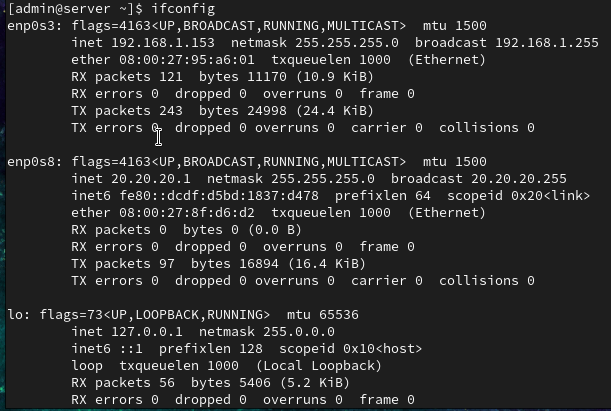


As we can see a new file called company.com.jnl is addesthat include dynamic addes records

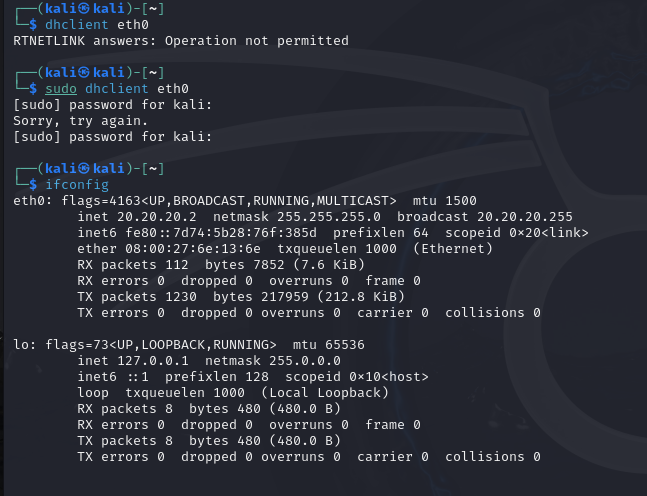
Also we can see that it add through the journalctl logs



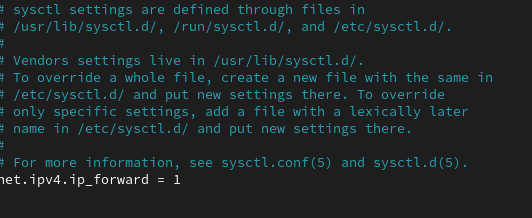
* Security tips:
* **DNS Hijacking**: Also known as domain theft, this involves maliciously gaining control of a domain name to redirect traffic to fake websites or launch other attacks.
* Solution: secure the dns by letting only the legitimate server and services to update the dns zone records, like setting a key for the dhcp and dns update flow
* **DNS Amplification Attack**: A form of DDoS attack where attackers exploit open DNS resolvers to flood a target server with traffic, amplifying the attack's strength
* **Solution**: allow only legitimate like internal network users to query the dns
* **DNS zone transfer Attack**: zone transfer is when a slave/secondary dns server ask the master/primary dns server for updating him with records in a zone, that can be don’t by a command line too, “dig -axfr ip-of-the-dns-that-has-the-target-zone”
* **Solution:** how slave servers can request zone transfer by TSIG “signature” feature of the bind service
* NAT
* Topology in virtualbox:
* Centos9 with bridge interface and internal interface, internal interface will be the linked interface to the LAN, while the bridge interface will be the public interface



* Kali will be the client with internal interface with an IP that it got it from the dhcp we set on the centos



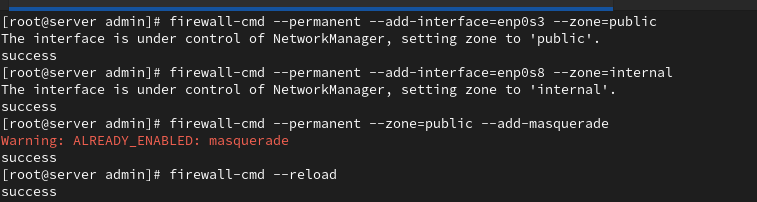
1. Enable IP forwarding by editing the /etc/sysctl.conf file and setting net.ipv4.ip\_forward = 1. This allows the CentOS system to forward packets between different network interfaces.



1. Reload the sysctl without rebooting the system



1. With firewalld-cmd: Add the interfaces to the appropriate zones, firewall-cmd --add-interface=public-interfce-name --zone=public and firewall-cmd --add-interface=internal-interface-name --zone=internal
2. Enable masquerading, which is an option that let the server replacing the source ip “internal private ip” in the packet by the public ip “the ip of the NAT interface” for the external zone: firewall-cmd --zone=public --add-masquerade –permanent reload the firwalld, firewalld-cmd –reload



1. Make sure that the kali client has the default getaway as the ip of the internal centos interface, we already set that when the dhcp set an IP to the kali machine, it set a dns IP and default getaway IP too with that

* NOTE: this is a virtual lab, the public internet wont get the kali request by the IP of the centos NAT’s IP, ofcourse it will be nated it self to your host machine private IP which its self will be translated to the public IP of your home network
* Now lets test if the kali has an internet access
* Pinging from the kali is working, now I entred youtub in tha kali machine, lets the /var/log/messages in the centos server

