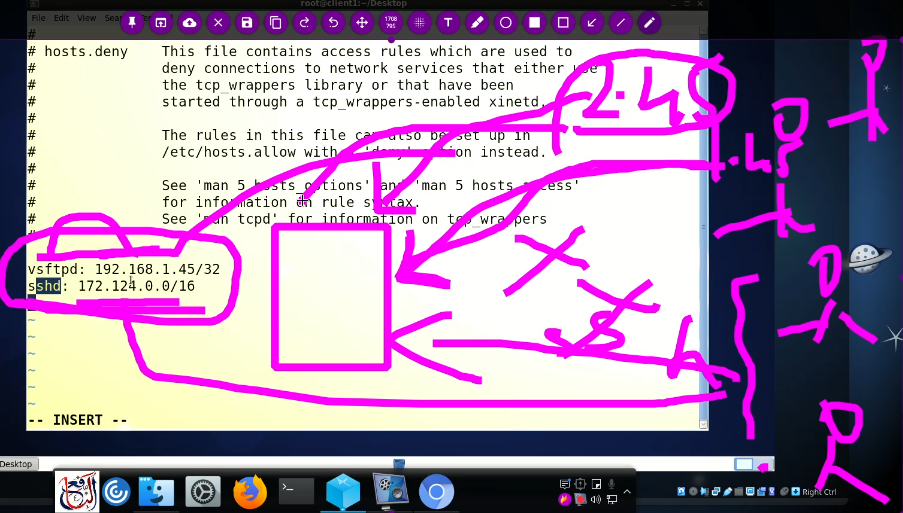
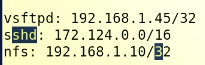
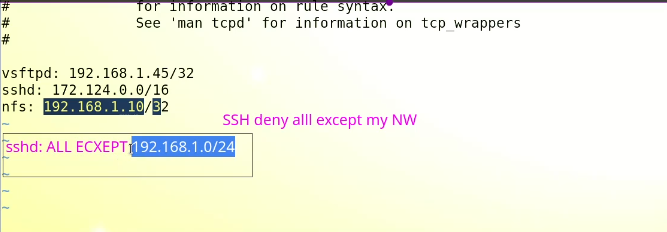
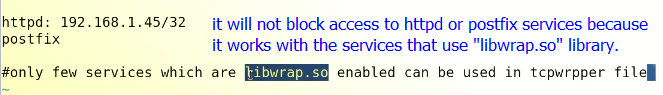
Lecture 14

**TCP Wrappers-IP Tables**

TCP Wrappers

It is **application-level firewall**,

It has two files,

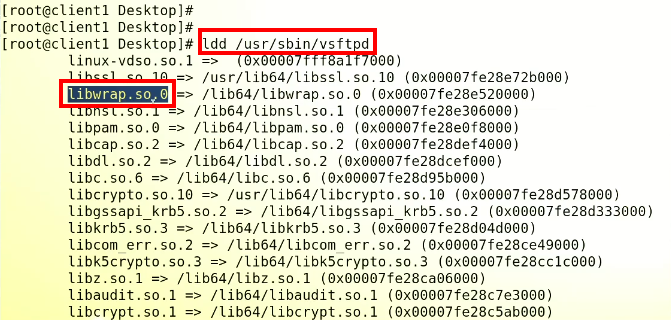
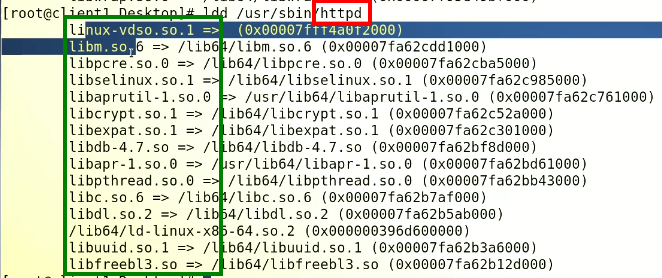
* .
  1. /etc/host.allow
  2. /etc/host.deny
* 
* All the requests from specified IPs will be denied as it is mentioned in /etc/host/deny file.
* Rules are defined as,
* 
* TCP wrapper is a software tool that provides network access control for Internet services running on a Unix or Linux operating system. It acts as a filter between the service and the client, intercepting incoming network connections and determining whether or not to allow the connection based on a set of rules.
* The main purpose of TCP wrapper is to enhance security by providing an additional layer of access control for network services. It can be used to restrict access to specific IP addresses or networks, and to monitor and log network connections.
* TCP wrapper works by intercepting incoming network connections and checking the source IP address of the client against a set of access rules defined in the configuration file /etc/hosts.allow and /etc/hosts.deny. If the client IP address matches an entry in the hosts.allow file, the connection is allowed. If it matches an entry in the hosts.deny file, the connection is denied.
* TCP wrapper is commonly used with network services such as SSH, FTP, Telnet, and SMTP to provide additional security and access control. It is also used in conjunction with intrusion detection systems and other security tools to provide a comprehensive security solution for Unix and Linux servers.
* Suppose there is a requirement which is “deny SSH to all except my network”.
* 
* Or the other way is to configure **host.allow** file and put network IP address there 🡪 **sshd: 192.168.1.0/24,**
* The rule **sshd: ALL EXCEPT 192.168.0.10/24** in the **/etc/hosts.deny** file means that **all hosts, except those in the range 192.168.0.10 to 192.168.0.254** (subnet mask 255.255.255.0), **are denied** SSH access to the system running the SSH daemon (**sshd**).
* **Interview Question** 🡪 if ALL is set in both host.allow & host.deny 🡪 the answer is in that case **“allow”** will take precedence. It works on **First-in-First-out method.**
* **.**

The **libwrap.so** library, also known as TCP Wrapper, is a software library that provides a simple access control mechanism for network services running on a Linux system. It is commonly used to restrict access to network services, such as SSH and Telnet, based on the IP address of the connecting host.

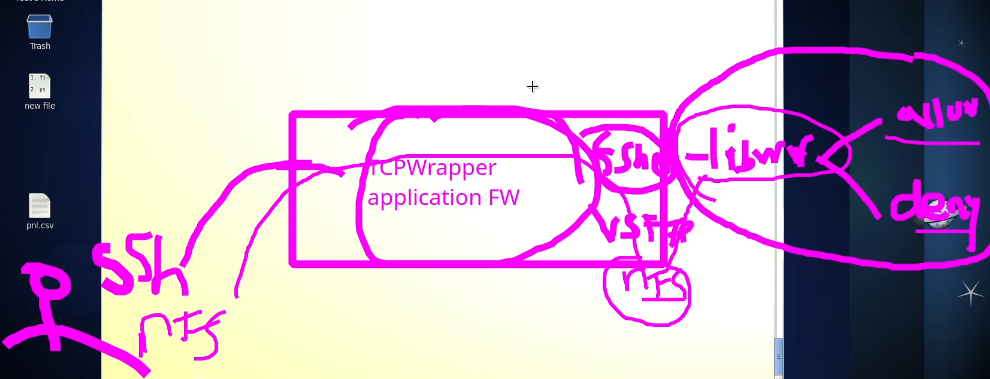
Some of the network services that use the **libwrap.so** library include:

1. **SSH:** The SSH daemon (**sshd**) is commonly configured to use TCP Wrapper to restrict access to SSH connections based on the source IP address. This is done by configuring the **sshd** daemon to use **libwrap.so** in its **/etc/hosts.allow** and **/etc/hosts.deny** configuration files.
2. **Telnet:** The Telnet daemon (**telnetd**) can also be configured to use TCP Wrapper to restrict access to Telnet connections based on the source IP address. This is done by configuring the **telnetd** daemon to use **libwrap.so** in its **/etc/hosts.allow** and **/etc/hosts.deny** configuration files.
3. **FTP:** The File Transfer Protocol (FTP) daemon (**vsftpd**) can also be configured to use TCP Wrapper to restrict access to FTP connections based on the source IP address. This is done by configuring the **vsftpd** daemon to use **libwrap.so** in its **/etc/hosts.allow** and **/etc/hosts.deny** configuration files.
4. **Mail servers:** Mail servers, such as Sendmail and Postfix, can also be configured to use TCP Wrapper to restrict access to incoming mail based on the source IP address. This is done by configuring the mail server software to use **libwrap.so** in its configuration files.

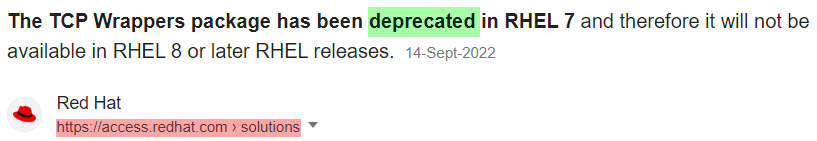
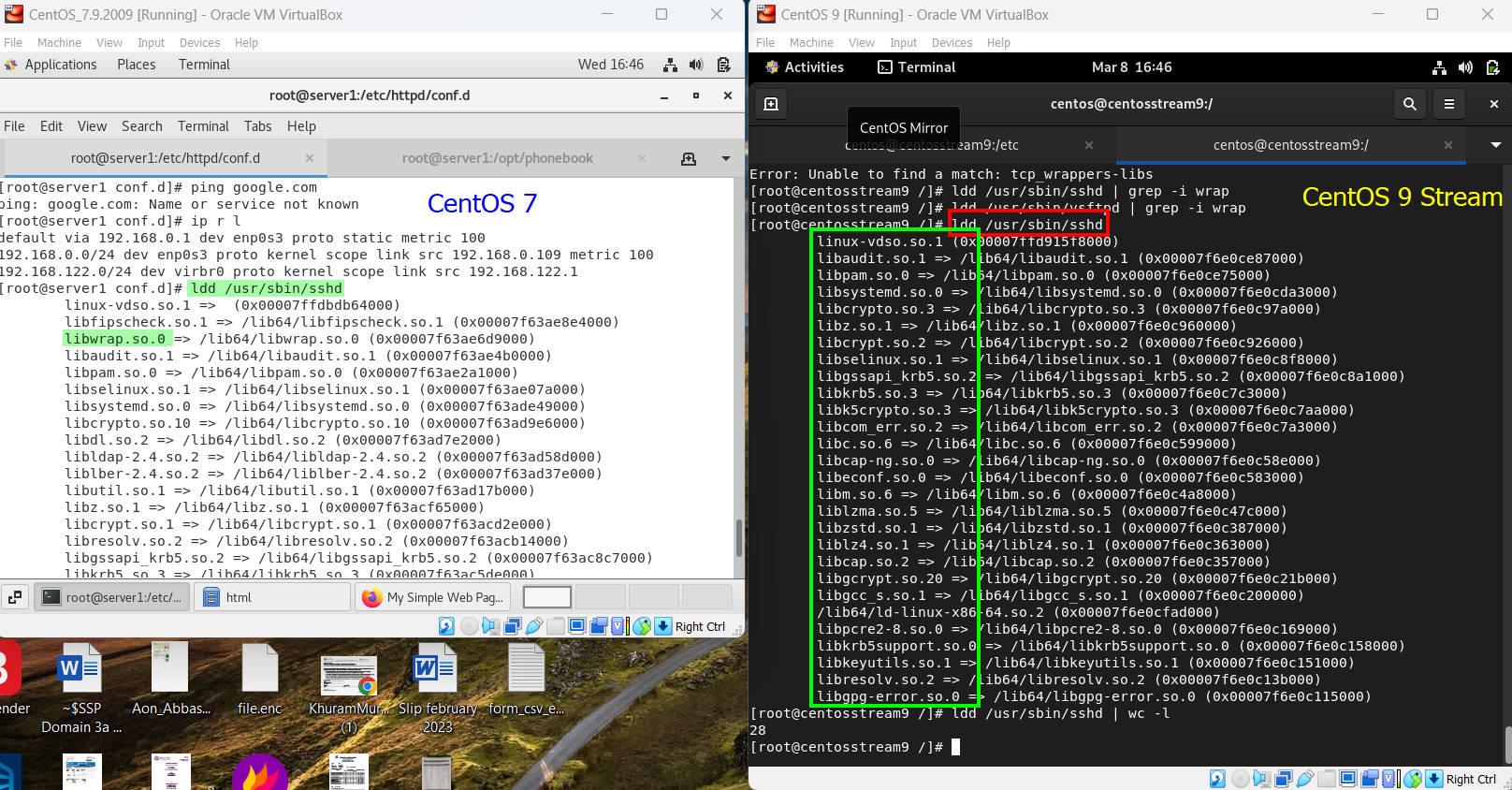
These are just a few examples of the network services that use the **libwrap.so** library to provide access control based on the source IP address of incoming network connections.

* **Shared object or .so library.**
* A binary file that uses **.so** libraries is a compiled executable file that depends on shared object libraries in the form of **.so** files. Shared object libraries are dynamic link libraries in Linux that contain code and data that can be loaded into memory at run-time and shared among multiple processes.
* When a binary file is compiled with a dependency on a **.so** library, it means that the binary file expects to find that library at runtime and dynamically link to it. This allows the binary file to be smaller and more efficient, as it does not need to include the entire code of the library in the binary file.
* For example, suppose we have a binary file named **my\_program**, and it depends on a shared object library named **libexample.so**. When **my\_program** is executed, it will attempt to load **libexample.so** into memory and link to it dynamically.
* The process of linking to a shared object library is handled by the dynamic linker, which is responsible for resolving the symbols needed by the binary file at runtime. This means that if the required **.so** library is not present on the system, the binary file will fail to execute and report a missing library error.
* In summary, a binary file that uses **.so** libraries is an executable file that depends on shared object libraries to provide functionality, which is dynamically linked at runtime.
* **What is a “binary”?.**
* In computing, a binary refers to a file that contains machine-readable code. It is an executable file format that can be executed directly by the computer's CPU. A binary file contains compiled code that can be executed by the operating system, without the need for the user to have access to the source code.
* Binary files can be used for a wide range of purposes, such as installing software, running programs, and storing data. For example, when you download and install an application on your computer, the installation package typically contains one or more binary files that are executed to install the application.
* Binary files are different from text files, which contain human-readable characters encoded in a specific character set. Binary files, on the other hand, contain machine code that is specific to the computer architecture and cannot be easily read or edited by humans.
* In summary, a binary file is a file that contains compiled code in a format that can be executed by the computer's CPU. It is an executable file format that is used for a wide range of computing tasks.
* Top of Form
* ***To check which services are dependent on “libwrap.so” library. Or any other library.***
* $ ldd /usr/bin.vsftpd
* **ldd** is a Linux command-line utility that is used to list the shared libraries that a binary executable depends on. The name **ldd** stands for **"list dynamic dependencies".**
* When you run the **ldd** command followed by the name of a binary executable, it will display a list of shared libraries that the executable depends on, along with the paths to those libraries. This can be useful when troubleshooting issues with an application or when you need to ensure that all the required libraries are present on a system.
* 
* It shows that the service “vsftpd” is dependent upon “libwrap.so” library.
* And its access can be controlled through ***TCP wrapper.***
* ***.***
* **If we check “sshd”**
* 
* If we check “httpd” with “ldd”command,
* 
* It shows there is no “libwrap.so” in the list.
* So, the TCPWrapper or “allow.deny” functionality can’t be applied on “httpd” or Apache services.

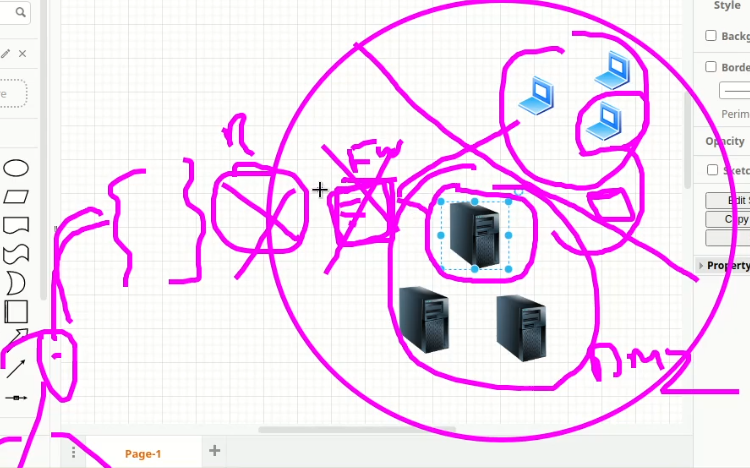
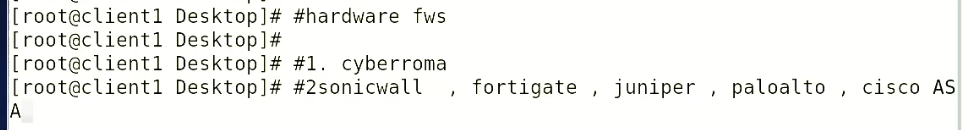
Working of **TCPWRapper**,

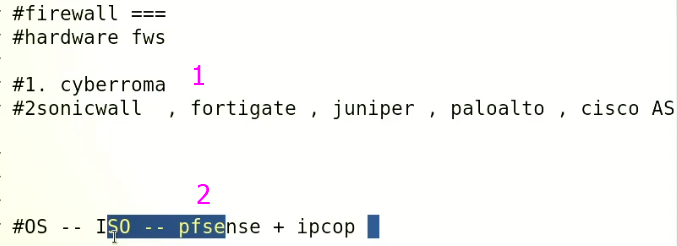
* 
* Server checks for “libwrap.so” library if the requested services is dependent on “libwrap.so” then it goes to “allow/deny” files to check the rule whether that particular IP is allowed or denied to use that specific service.
* If there was a Kernel Level Firewall 🡪 it would be a batter strategy.
* Tihs is called **IP Tables** (**firewalld**)

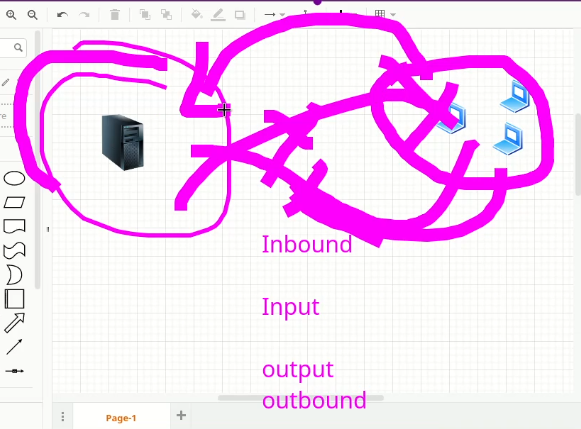
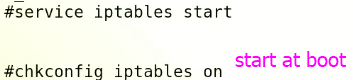
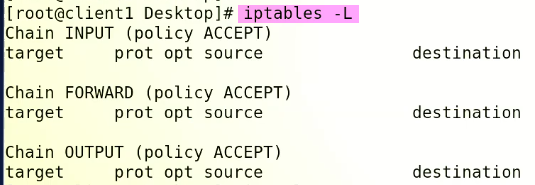
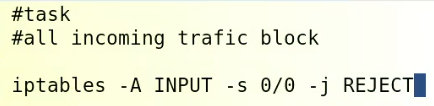
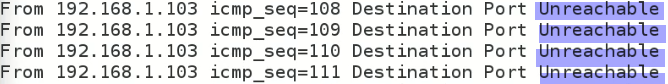
**Alert !!!**

* 
* .
* I searched for this post because 🡪 I tried **$ ldd /usr/sbin/sshd** on CentOS 9 Stream and I was surprised to see that there is no “libwrap.so” in the list. It increased my curiosity and I cross checked it on CentOS 7. It was available on CentOS 7.
* So I had a confusion whether it is still available in latest RHEL versions or not and I found above mentioned information.
* 
* .

**IP Tables**

* 
* .
* 
* .
* A picture containing electronics, drive, projector

  Description automatically generated
* .
* 
* .
* A picture containing graphical user interface

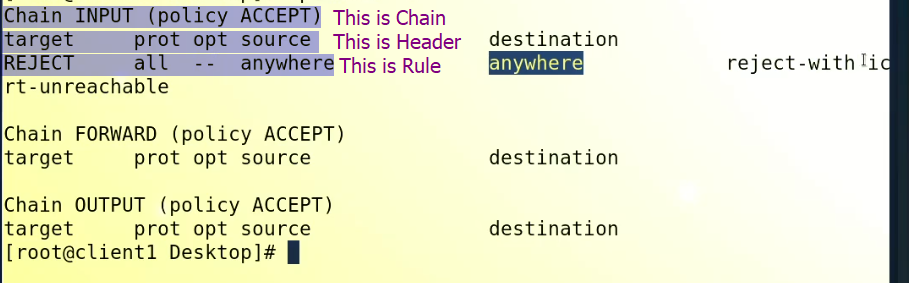
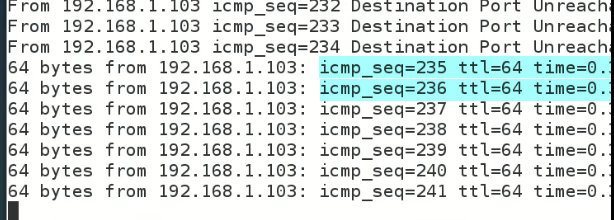
  Description automatically generated
* .
* **IP Table**
* IP tables is a Linux-based software firewall that allows you to configure and manage network traffic rules for IPv4 and IPv6 traffic. IP tables uses a set of rules to filter network packets and forward or block traffic based on various criteria such as source IP address, destination IP address, port number, protocol type, and more.
* IP tables is typically used to secure a server by blocking unwanted traffic and allowing only the necessary traffic to pass through. It can also be used to set up network address translation (NAT) and port forwarding to enable access to services running on a private network from the public internet.
* IP tables is command-line based and is part of the Linux kernel. It can be configured by editing the rules directly using the **iptables command**, or by using a graphical front-end such as Firestarter or UFW (Uncomplicated Firewall).
* IP tables has been replaced by *nftables* in some Linux distributions, but is still widely used and supported.
* Top of Form
* .
* 
* .
* 
* 
* .
* This is first command, **-L** *lists which rules are configured currently.*
* .
* Task is to block all incoming traffic.
* Right now, CentOS 6 machine can PING by CentOS 7
* We need to stop it,
* 
* . with the execution of this command have blocked all inbound traffic and
* 
* PING is stopped at CentOS 7
* 
* The command "**iptables -A INPUT -s 0/0 -j REJECT**" is an example of an iptables command in Linux.
* This command adds a new rule to the INPUT chain of the iptables firewall. The rule specifies that any incoming traffic with a source address of "**0/0**" (i.e., any IP address) should be rejected, meaning that it will not be allowed to pass through the firewall.
* The **"-j REJECT"** option specifies that any traffic matching this rule should be rejected with an ICMP error message indicating that the traffic is not allowed.
* Overall, this rule is useful for blocking all incoming traffic to the system, which can be a useful security measure in certain situations. However, it is important to carefully consider the implications of such a rule before implementing it, as it may interfere with legitimate traffic that you want to allow through the firewall.

*iptables -A INPUT -s 0/0 -j REJECT this rule didn't blocked traffic from CentOS 7*

***I restarted the VMS and it worked perfectly.***

If the rule "iptables -A INPUT -s 0/0 -j REJECT" did not block traffic from CentOS 7, there could be several reasons why:

1. The rule was not added to the firewall correctly: Make sure that the rule was added correctly using the correct syntax and that there were no errors. You can use the "iptables -L" command to list all the rules in the firewall and check if the rule was added.
2. The rule was added after the traffic was already allowed: If the traffic was allowed before the rule was added, then the rule will not block the traffic. In this case, you may need to restart the firewall or flush the existing rules and add the new rule again.
3. The traffic was not coming from the expected source: If the traffic was not coming from "0/0" as specified in the rule, then the rule will not match the traffic and it will be allowed through the firewall. In this case, you may need to update the rule to match the correct source of the traffic.
4. The traffic was allowed by a previous rule: If there is another rule in the firewall that allows the traffic, then the traffic will be allowed through the firewall even if this rule is added. In this case, you may need to review all the rules in the firewall and make sure that there are no conflicting rules.

* 
*  🡪 it is called “Flush” 🡪 rule ko nikal dena
* With this command the Connectivity with the server is resumed.
* 
* CHAIN is most important part in IP Tables
* Coming traffic is called INPUT
* The outgoing traffic is called OUTPUT
* The traffic which comes in and goes to another machine is called FORWARD (it requires 2 Network Cards)
* Same command can be set just replace INPUT with OUTPUT
* Rules stores in RAM (temporarily)
* To make it permanent,

$ service iptables save

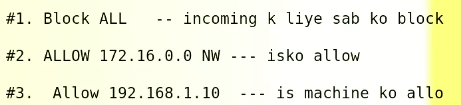
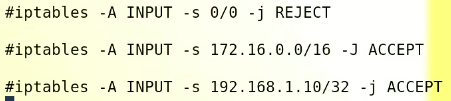
* 
* Changes are save in **/etc/sysconfig/iptables**
* To change rule for **OUTPUT**
*  🡪 -d means destination
* Working of IP Tables is explained as,
* .
* .
* .

**iptables -A INPUT -s 0/0 -j REJECT**

* "-A" option: This option specifies that a new rule should be **appended** to the end of the specified chain. In this case, the chain is the "INPUT" chain, which is used to filter incoming packets.
* "INPUT": This specifies the chain in which the rule should be added. The "INPUT" chain is used to filter incoming packets.
* "-s 0/0": This specifies the **source** IP address or IP range that the rule should match. "0/0" is a shorthand for **any IP address**, meaning that the rule will match any packet regardless of its source IP address.
* **"-j REJECT"**: This specifies the action that should be taken if a packet matches the rule. In this case, the action is to reject the packet and send an ICMP error message back to the sender indicating that the packet was rejected. The "**-j**" option is used to specify the target for the rule, which in this case is the "**REJECT**" target.
* *Sir Kazim explained that -j means “jump”*
* *You are correct. The "-j" option in the "iptables" command stands for "jump" and is used to specify the target that should be jumped to if a packet matches the rule. In the case of "iptables -A INPUT -s 0/0 -j REJECT", the target specified is "REJECT", which means that if a packet matches the rule, it will be rejected and a rejection message will be sent back to the sender of the packet. So, the "-j" option is used to jump to the specified target when a packet matches the rule. Thank you for catching that mistake in my previous response.*

Overall, this command adds a new rule to the "INPUT" chain of the iptables firewall, which matches any incoming packet from any source IP address and rejects it with an ICMP error message.

Hands on Practice

* 
* Solution
* 
* It should work but ,
* It will reject all because 1st RULE is set and the system will ignore the 2nd and 3rd rules.
* The order should be as below,
* 