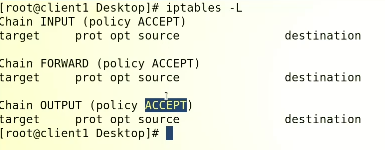
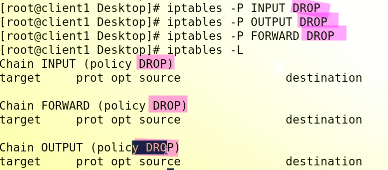
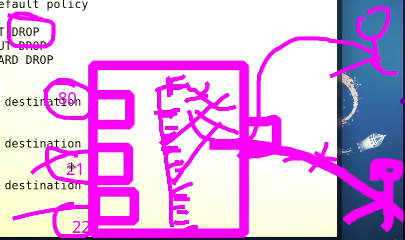
Lecture 15

**IPTables-FirewallD-SELinux**

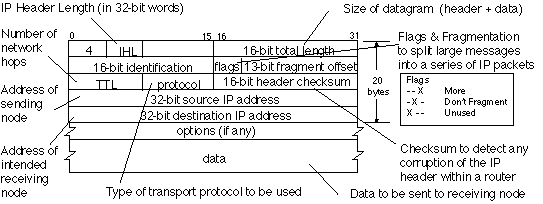
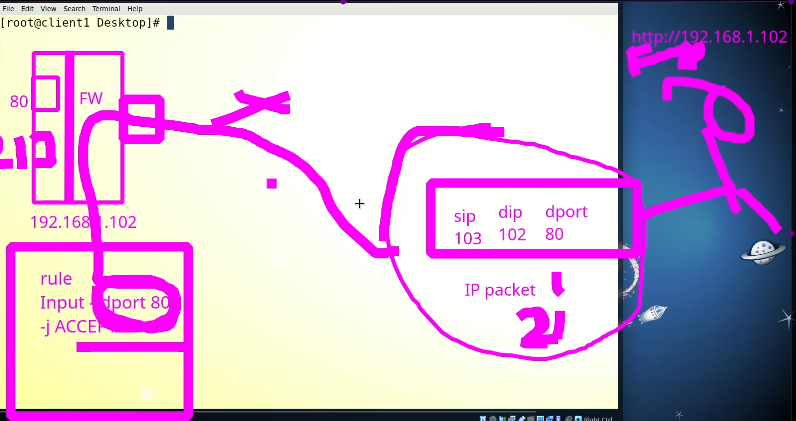
IP Tables

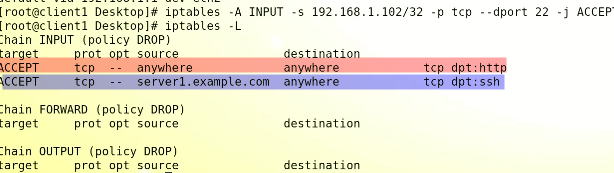
* $ iptables -L
* 
* By default, everything is set to **“Accept”**
* But it is not industry standard,
* In industry Hardware firewalls are used

How to change default policy?

* 
* It will block all traffics,
* 
* But it’s not recommended solution,
  + We will add the rules One by one as per requirement.

**Task** (How to open ports?)

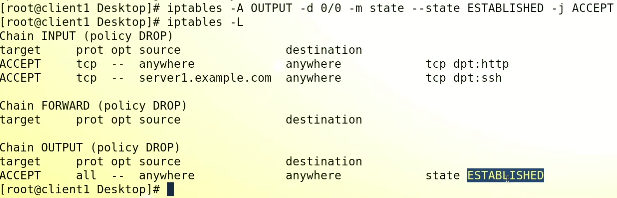
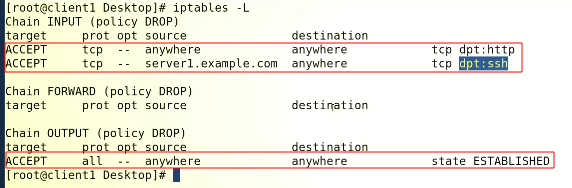
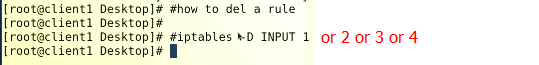
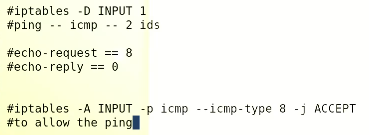
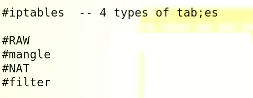
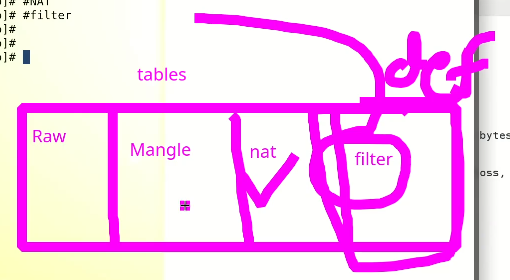
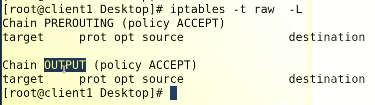
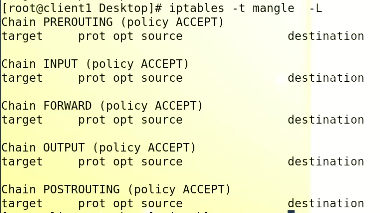
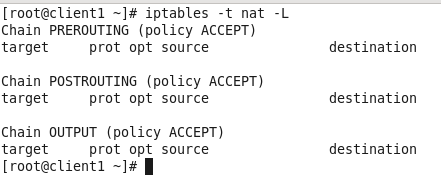
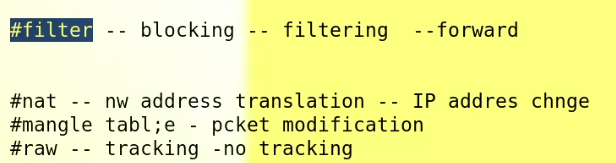
* Give access to port 80 and block for all other ports.
* IP Packet
* 
* The IP (Internet Protocol) packet architecture is a set of rules and conventions that govern the format and contents of an IP packet, which is used to transmit data across the internet.
* An IP packet typically consists of two main parts: the **header** and the **payload**.
* The header contains information about the **source** and **destination** of the packet, as well as other metadata that is used by network devices to route and deliver the packet. The header is further divided into several fields, including the version, length, type of service, identification number, time to live, protocol, checksum, and source and destination IP addresses.
* The payload is the actual data being transmitted, such as a file, a web page, or an email message.
* The IP packet architecture is designed to be flexible and extensible, so that new features and functionality can be added as needed. For example, newer versions of the IP protocol, such as IPv6, include additional fields in the header to support larger address spaces and more efficient routing.
* 
* The rule will be set according to IP Packet
* $ iptables -A INPUT -s 0/0 - -p tcp - -deport -j ACCEPT
* Graphical user interface, application

  Description automatically generated with medium confidence
* **Task**
* 
* **What is TCP?**
* TCP (Transmission Control Protocol) is a standard communication protocol used in computer networks, including the Internet. TCP is a connection-oriented protocol, which means that before data can be exchanged between two devices, a connection must be established between them.
* *TCP provides reliable, ordered, and error-checked delivery of data between applications running on different devices. It is responsible for breaking data into packets, numbering them, and sending them to the recipient device in the correct order. The recipient device acknowledges receipt of the packets and sends back an acknowledgement to the sender. If any packets are lost or corrupted during transmission, TCP will automatically retransmit them until they are received correctly.*
* TCP also performs flow control, which means it adjusts the rate of data transmission based on the receiving device's capacity to handle the data. This helps prevent data overload and ensures that the recipient device can process the data effectively.
* TCP is widely used in many applications that require reliable data transfer, such as web browsing, email, file transfers, and streaming media.
* ***There is problem with this rule*** because it will only allow incoming access but blocks outgoing access,
  + This rule will not establish connection to source node or machine.
  + The firewall’s OUTPUT chain will block outgoing access.

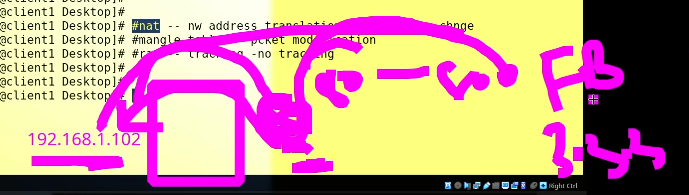
**Remember!!!**

* + there is a concept in firewall that the connection which is established should be allowed to go out.
  + States in firewall,
    - Established
    - New
    - realted

**SOLUTION**

* **$ iptables OUTPUT -A -d 0/0 -m state - - established -j ACCEPT**
* 
* .
* 
* Now the server is fully **secured**.
* .
* 
* .
* **Ping** is also not working because it is “icmp” protocol which is not allowed by firewall. That’s why firewall is blocking it.
* 
* To allow PING
* 
* There r 4 types of tables in IP Tables
* 
* By default rules are written in “filter” table
* 
* .
* To watch chains in other tables,
* $ iptables -t raw -L
* 
* .
* 2.
* 
* 3.
* 
* .
* 
* .

**NAT Table**

* 
* With private IP (192.168.1.102) we cant go to internet, the router assigns its Public IP to serve the internet. It sometimes changes the port No. too. This is called Network Address Translation.

*Network Address Translation (NAT) is a technology used to translate IP addresses between networks. It allows multiple devices in a private network to share a single public IP address to access the internet.*

Here's how NAT works:

1. Private IP addresses: Devices in a private network are assigned private IP addresses, which are not routable on the internet. Private IP addresses are used to create a unique identifier for each device on the local network.
2. Public IP address: The router or gateway device that connects the private network to the internet is assigned a public IP address by the internet service provider (ISP). This public IP address is used to identify the network on the internet.
3. Translation: When a device in the private network sends a packet to the internet, the router replaces the private IP address with the public IP address of the router. This translation is stored in a translation table in the router.
4. Response: When the internet responds with a packet to the public IP address of the router, the router looks up the translation table to determine which private IP address the packet should be forwarded to.
5. Port translation: To allow multiple devices to use the same public IP address, the router also translates the port number in the packet header. The router assigns a unique port number to each device on the private network, and when a response is received, it uses the port number to forward the packet to the correct device.

NAT allows a private network to use a single public IP address, which conserves the limited number of public IP addresses available. It also provides a level of security by hiding the private IP addresses of devices on the local network from the internet.

* .

**“firewalld”.** 🡪 CentOS 7 & 8 and onward

firewalld is a firewall management tool used in Linux operating systems to control inbound and outbound traffic to and from the system. It replaces the previous iptables firewall and provides a more user-friendly interface for managing firewall rules.

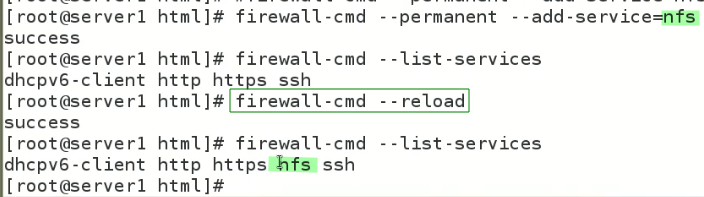
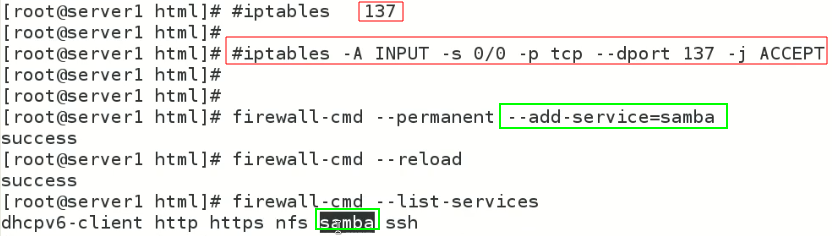
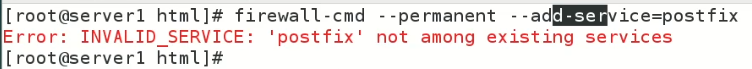
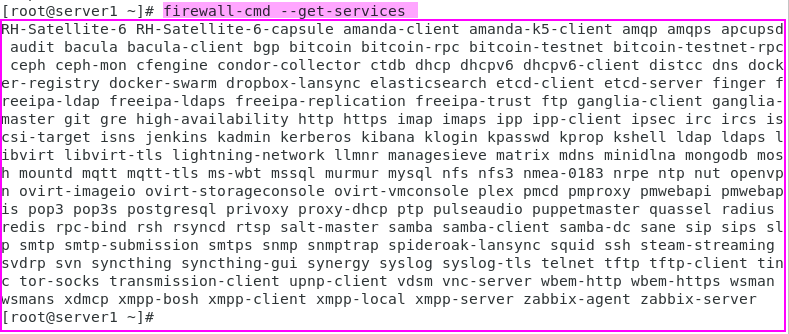
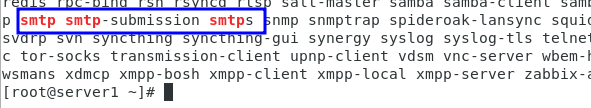
*firewalld works by creating a set of predefined rules that define how traffic is allowed or blocked on the system. These rules are organized into zones, which define different levels of trust for different network connections. By default, firewalld comes with several pre-configured zones, such as public, internal, and trusted.*

firewalld uses a dynamic firewall model, which means that rules can be added or removed without having to restart the firewall. This allows for more flexibility in managing firewall rules and reduces the risk of disrupting network connections.

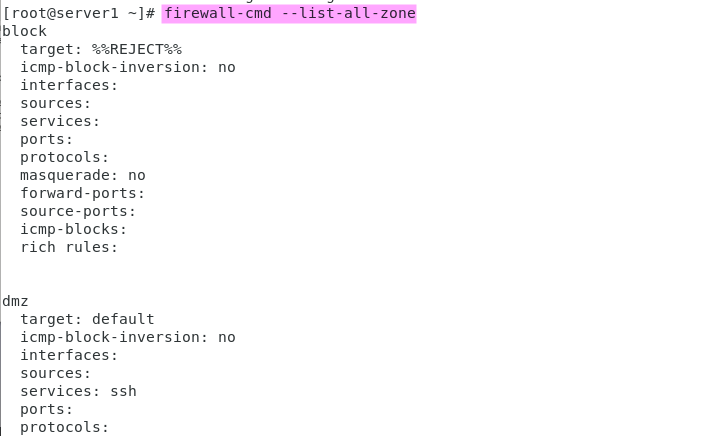
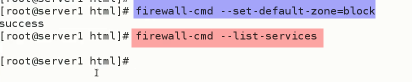
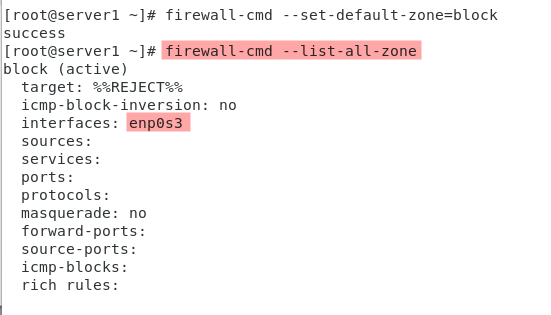
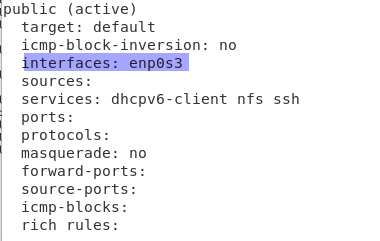
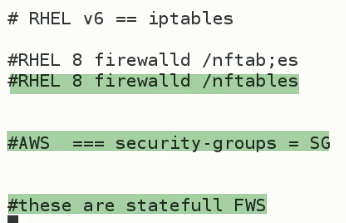
In firewalld, rules are defined using services, ports, and protocols. A service is a collection of one or more ports and protocols that are used to identify a particular network application or service. For example, the httpd service includes port 80 for HTTP traffic. A port is a specific number used to identify a network connection, while a protocol defines the format and structure of the data being transmitted over the connection.

firewalld provides several command-line tools and graphical user interfaces (GUI) to manage firewall rules. The most commonly used tool is the firewall-cmd command, which allows you to add, remove, and modify firewall rules. Other tools include firewall-config, which is a graphical interface for managing firewalld, and firewall-applet, which is a tray icon that displays the firewall status and allows you to enable or disable it.

Overall, firewalld provides a flexible and user-friendly interface for managing firewall rules in Linux systems. It simplifies the process of managing firewall rules and provides a more secure environment for network connections.

* ***$ systemctl status firewalls***
* ***$ firewall-cmd --state***
* Here tables are called “zones”.
* Here default is **“public zone”.**
* **How to open ports or services?**
* To check,
* 
* To open a ***port*** in "firewalld”
* 
* To make the changes take effect,
* 
* 
* *How to add port in firewalld as compared to iptables*
* 
* .
* If we want to add “postfix” service (email service)
* 
* It throws error,
* ***Actually, firewalld maintains a database which lists the services.***
* 
* 
* .

Firewalld uses “zones”

* $ firewalld-cmd --get-zones
* 
* To list all zones and rules, (in detail)
* 
* .
* How to block all?
* By default public zone is active,
  + The solution is to change the default zone to “block zone” .
* **$ firewall-cmd -- set - default-zone-block**
* 
* If we list all zones, there is noting to display because everything is blocked.
* 
* *The interfaced (enp0s3) if listed in block zone and all the traffic through this interface is blocked.*
* If “public” zone is selected back, 🡪 the same interface is listed in public zone and removed from “block zone”
* 
* 
* *In AWS stateless firewall is called NACL(Network Access Control List) 🡪 and Security Group is “stateful firewall in AWS”*
* Sir Kazim told another difference 🡪 in CentOS 6 in iptables **(stateless firewall)** if connection is established, we need to set both INBOUN (incoming ports) and OUTBOUND (outgoing ports) rules for a connection to work properly, but in **“firewalld”** there is no need to write rules like old one.
* A **stateful firewall** is a type of firewall that monitors and tracks the state or context of network connections passing through it. Unlike stateless firewalls, which only inspect each packet in isolation, a stateful firewall is able to examine the overall context of a connection to make more informed decisions about whether to allow or block traffic.
* The stateful firewall maintains a record of each connection's state, including information such as source and destination IP addresses, port numbers, and sequence numbers. This information is used to determine if incoming packets are part of an existing connection or if they are attempting to initiate a new connection. By keeping track of these details, a stateful firewall can detect and block suspicious traffic, such as packets with incorrect sequence numbers or those originating from untrusted sources.
* Overall, stateful firewalls are considered to be more effective and secure than stateless firewalls because they are able to provide a higher level of inspection and control over network traffic.

**SELinux**

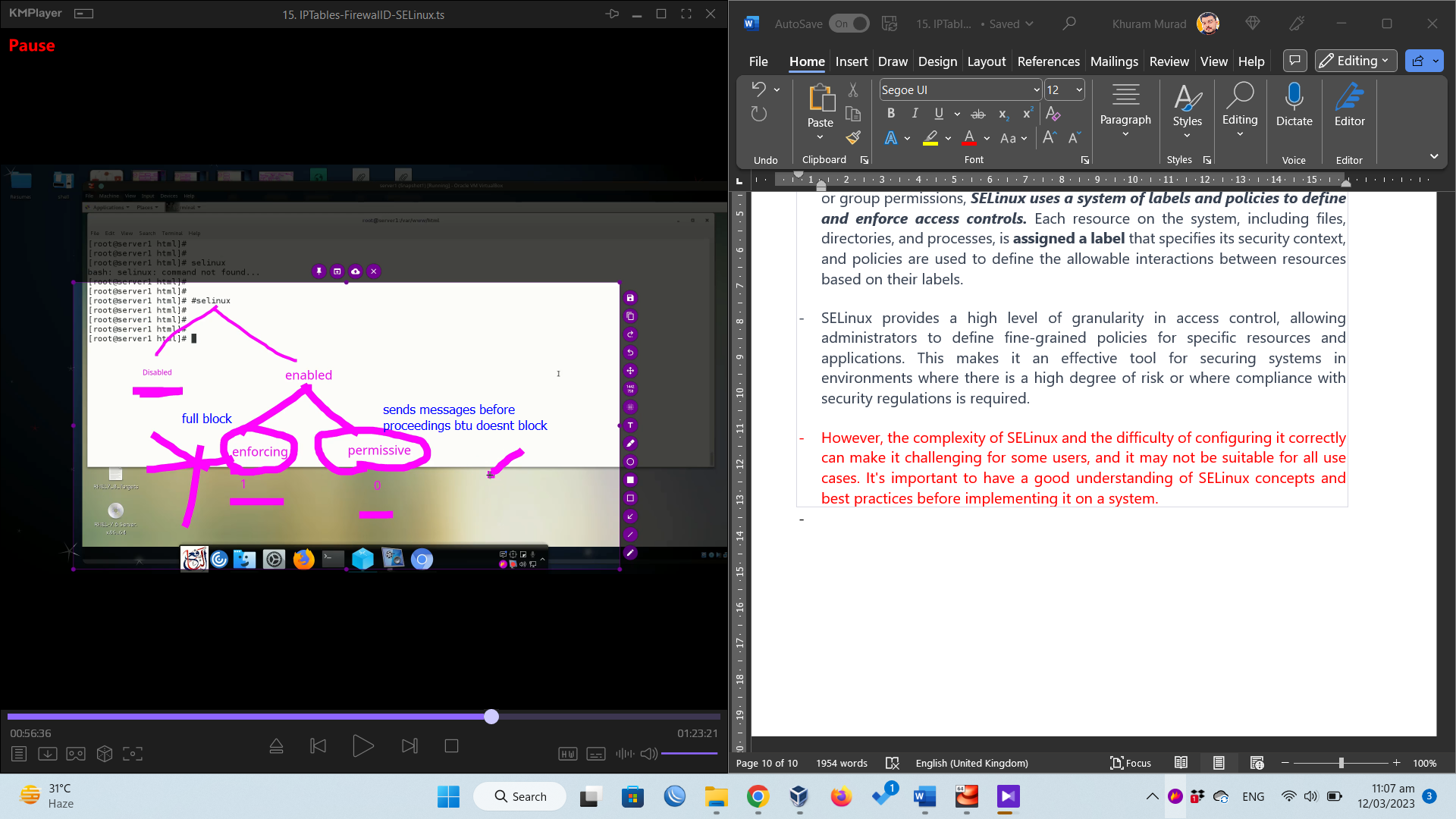
* Security Enhance Linux
* It is OS level security
* Linux is already secured but SELinux is an additional security layer.
* SELinux stands for Security-Enhanced Linux, which is a security feature that is built into many Linux operating systems. SELinux provides a set of access control mechanisms that can be used to enforce **mandatory access controls** **(MAC)** on system resources, such as files, directories, and processes.
* Unlike traditional Linux access controls, which are discretionary and rely on user or group permissions, ***SELinux uses a system of labels and policies to define and enforce access controls.*** Each resource on the system, including files, directories, and processes, is **assigned a label** that specifies its security context, and policies are used to define the allowable interactions between resources based on their labels.
* SELinux provides a high level of granularity in access control, allowing administrators to define fine-grained policies for specific resources and applications. This makes it an effective tool for securing systems in environments where there is a high degree of risk or where compliance with security regulations is required.
* However, the complexity of SELinux and the difficulty of configuring it correctly can make it challenging for some users, and it may not be suitable for all use cases. It's important to have a good understanding of SELinux concepts and best practices before implementing it on a system.
* .

**use cases.**

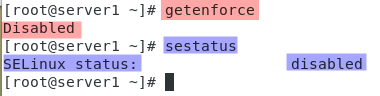
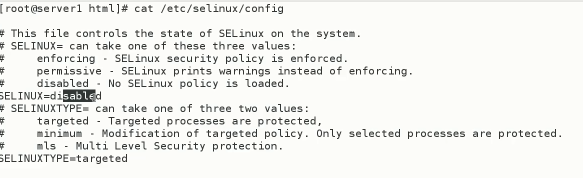
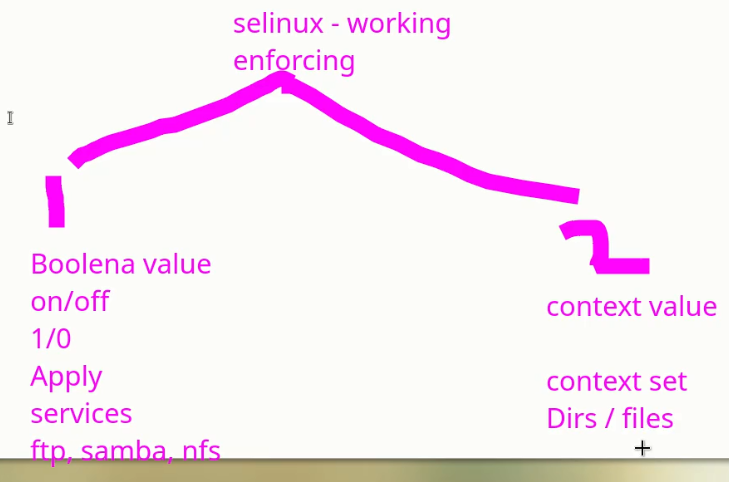
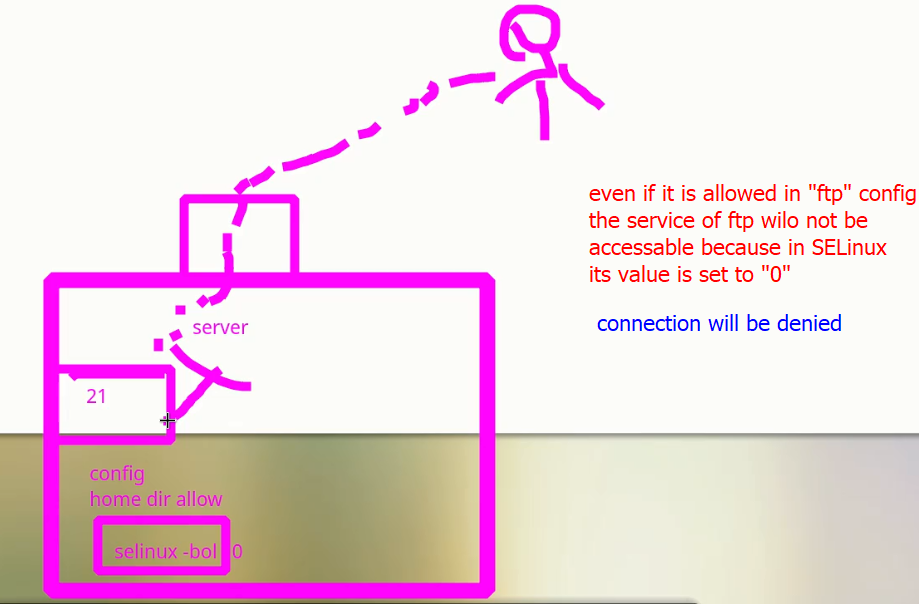
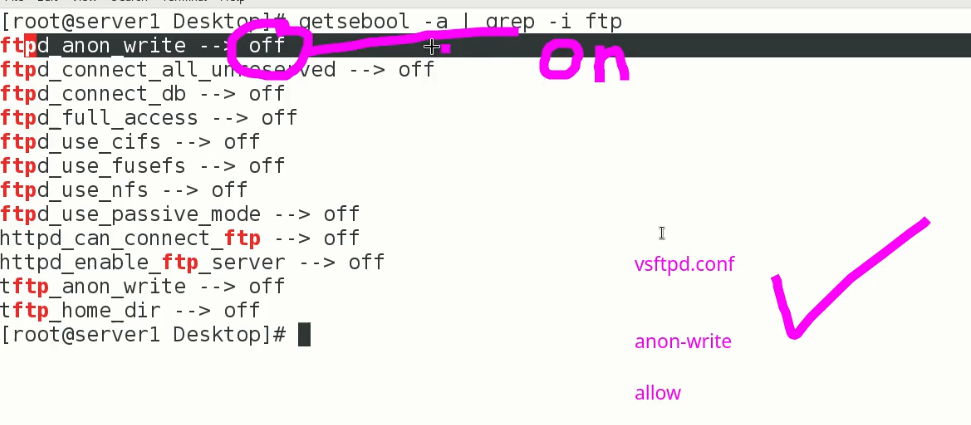
Here are some use cases where SELinux can be useful:

1. **Mandatory Access Control**: SELinux provides a robust mandatory access control (MAC) system that enforces policies on how different processes and users can access resources on the system. This feature can help reduce the risk of unauthorized access and improve the security of the system.
2. **Hardened Applications**: SELinux can be used to harden applications by restricting their access to resources they do not require. By defining custom policies for applications, it is possible to limit the impact of potential security vulnerabilities in the software.
3. **Multi-Tenancy**: SELinux can be used to isolate multiple tenants or customers on a shared system. By defining policies for each tenant, it is possible to ensure that they can only access their own data and resources, while preventing them from accessing other tenants' data.
4. **Compliance**: SELinux can help organizations comply with regulatory requirements by providing a means to enforce security policies on systems that process sensitive information. By using SELinux, organizations can demonstrate that they have implemented appropriate security controls.
5. **Containerization**: SELinux can be used to improve the security of containerized environments by restricting the access of containerized processes to only the resources they require. This helps to limit the impact of any security vulnerabilities in the container.
6. **Cloud Security**: SELinux can help improve the security of cloud-based systems by providing an additional layer of security controls. By defining policies for cloud instances and services, it is possible to ensure that they are protected from unauthorized access and attacks.

Overall, SELinux is a powerful tool for improving the security of systems and applications, and it can be used in a wide variety of use cases to provide a secure and reliable computing environment.

* 
* .
* From “enforcing” to “permissive” 🡪 no reboot
* From “permissive” to “enforcing” 🡪 no reboot
* From “enabled” to “disabled” 🡪 reboot
* From “disabled” to “enabled” 🡪 reboot

1. From "enforcing" to "permissive" -> no reboot required: When changing from "enforcing" mode to "permissive" mode, you do not need to reboot the system. In permissive mode, SELinux will log policy violations but will not actually enforce them, allowing you to see how SELinux would affect your system without causing any disruption.
2. From "permissive" to "enforcing" -> no reboot required: Similarly, when changing from "permissive" mode to "enforcing" mode, you do not need to reboot the system. In enforcing mode, SELinux will actively enforce policy violations, so it's important to ensure that your system is configured correctly before making this change.
3. From "enabled" to "disabled" -> reboot required: If you want to disable SELinux altogether, you will need to reboot the system after making the change. This is because SELinux is a core part of the Linux security model, and disabling it requires changes to the kernel that cannot be made without a reboot.
4. From "disabled" to "enabled" -> reboot required: Similarly, if you want to enable SELinux after it has been disabled, you will need to reboot the system to apply the changes.

* When SELinux is **enabled**, it means that the security module is actively running and enforcing mandatory access controls (MAC) on system resources, such as files, directories, and processes. This provides an additional layer of security on top of traditional Linux access controls and can help to protect the system from security threats.
* On the other hand, when SELinux is **disabled**, it means that the security module is not running and not enforcing any access controls. This can leave the system vulnerable to security threats, as it relies solely on traditional Linux access controls, which are discretionary and can be overridden by users with appropriate permissions.
* Check status of SELinux
* $ sestatus
* $ getenforce
* To change
* $ setenforce 0 🡪
* 
* **$ etc/selinux/config 🡪 for configuration**
* 
* **Working of SELinux explained.**
* 
* **.**
* **Services** *(how to set SELinux on services)*
* .
* 
* To change SELinux settings,
* $ setsebool -P <service>
* 

**Context Value (for directories and files)**

* **$ ls -Za**
* The command "ls -Za" lists the contents of the current directory in a long format that includes file permissions and additional information, including any special security context or SELinux context that is set for each file. Here's an example of what the output might look like:
  + drwxr-xr-x. 2 user user system\_u:object\_r:user\_home\_t:s0 4096 Mar 12 15:22 .
  + drwxr-xr-x. 18 user user system\_u:object\_r:user\_home\_t:s0 4096 Feb 28 11:07 ..
  + -rw-r--r--. 1 user user unconfined\_u:object\_r:user\_home\_t:s0 1108 Mar 12 15:22 example.txt
* In this example, you can see that there are two directories (denoted by the "d" in the first column), and one file ("example.txt") in the current directory. The file permissions are listed in the columns following the first one, and the security context information is listed in the third column.
* .
* 
* The string "system\_u:object\_r:public\_content\_t:s0" is an SELinux context that is associated with the directory "/var/ftp/".
* In SELinux, each object (such as a file or directory) has a security context that defines its access controls and other security attributes. The security context is made up of several parts, including a user identifier, a role identifier, a type identifier, and a sensitivity label.
* The "system\_u" part of the security context identifies the SELinux user that the object belongs to. The "object\_r" part identifies the role associated with the object. The "public\_content\_t" part identifies the type of the object, which in this case is "public\_content\_t". This type is used to identify directories that contain files that can be shared publicly. Finally, the "s0" part is the sensitivity label for the object.
* So in summary, the security context "system\_u:object\_r:public\_content\_t:s0" associated with "/var/ftp/" indicates that the directory contains files that can be shared publicly.
* .
* **Different identifiers of SELinux are explained**

different identifiers in SELinux

In SELinux, each object (such as a file, directory, or process) is associated with a security context, which includes several identifiers that determine its access controls and other security attributes. Here are the different identifiers used in SELinux:

1. **User Identifier (UID)**: This identifier specifies the SELinux user account that the object is associated with. Users are defined by SELinux policy and typically map to system user accounts or roles.
2. **Role Identifier (RID)**: This identifier specifies the SELinux role that the object is associated with. Roles define the types of operations that an SELinux user can perform on an object. Users can be assigned to multiple roles.
3. **Type Identifier (TID)**: This identifier specifies the SELinux type that the object is associated with. Types define the security attributes and access controls for the object. For example, a file type might indicate whether the file is executable, writable, or readable.
4. **Level Identifier (LID)**: This identifier specifies the SELinux sensitivity label that the object is associated with. Sensitivity labels define the security level or classification of the object, such as "top secret" or "unclassified".

Together, these identifiers form the security context for an object. The SELinux policy defines rules for how objects with different security contexts can interact with each other, which helps to enforce a more secure system.

* Examples ,

Here is a more comprehensive list of possible values for the SELinux identifiers along with examples:

1. **User Identifier (UID)**:
   * **system\_u**: Default SELinux user for system processes and services.
   * **user\_u**: Default SELinux user for regular user accounts.
   * **staff\_u**: SELinux user for administrative users.
   * **guest\_u**: SELinux user for guest users.
   * **root**: SELinux user for the root account.
   * **unconfined\_u**: SELinux user for users without any confinement.
2. **Role Identifier (RID)**:
   * **object\_r**: Default SELinux role for objects such as files and directories.
   * **staff\_r**: SELinux role for administrative users.
   * **sysadm\_r**: SELinux role for system administrators.
   * **user\_r**: SELinux role for regular users.
   * **guest\_r**: SELinux role for guest users.
3. **Type Identifier (TID)**:
   * **file**: Default SELinux type for regular files.
   * **dir**: Default SELinux type for directories.
   * **process**: SELinux type for processes.
   * **socket**: SELinux type for network sockets.
   * **chr\_file**: SELinux type for character devices.
   * **blk\_file**: SELinux type for block devices.
   * **fifo\_file**: SELinux type for FIFOs or named pipes.
   * **lnk\_file**: SELinux type for symbolic links.
4. **Level Identifier (LID)**:
   * **s0**: Default SELinux sensitivity label for most objects, indicating that they are unclassified or have no specific sensitivity level.
   * **s1**, **s2**, **s3**: Sensitivity labels used for objects that require increasing levels of security clearance.
   * **topsecret**: Sensitivity label used for objects that require the highest level of security clearance.
   * **system\_low**, **system\_medium**, **system\_high**: Sensitivity labels used for system-level objects with different security levels.

Here are some examples of security contexts that use these identifiers:

* **system\_u:object\_r:file\_t:s0**: Context associated with a regular file owned by a system process or service.
* **user\_u:object\_r:user\_home\_t:s0**: Context associated with a file or directory owned by a regular user.
* **staff\_u:staff\_r:admin\_t:s0**: Context associated with a file or directory owned by an administrative user with additional permissions.
* **system\_u:system\_r:httpd\_t:s0**: Context associated with the Apache HTTP server process.
* **system\_u:object\_r:net\_conf\_t:s0**: Context associated with network configuration files.
* **unconfined\_u:unconfined\_r:unconfined\_t:s0-s0:c0.c1023**: Context associated with an unconfined user account that is not subject to SELinux policy restrictions.

**commands associated with SELinux**

show all commands and all flags with examples

Here are some SELinux commands along with their possible flags and examples:

1. **sestatus**: This command displays the current status of SELinux.

* **-v**: Displays more detailed information about the SELinux status.
  + Example: **sestatus -v**
* **-b**: Displays the SELinux policy capabilities of the kernel.
  + Example: **sestatus -b**
* **-p**: Displays the current SELinux policy.
  + Example: **sestatus -p**

1. **getenforce**: This command displays the current SELinux mode.

* No flags are available for this command.
  + Example: **getenforce**

1. **setenforce**: This command is used to change the SELinux mode.

* **0**: Sets SELinux to permissive mode.
  + Example: **setenforce 0**
* **1**: Sets SELinux to enforcing mode.
  + Example: **setenforce 1**

1. **semanage**: This command is used to manage SELinux policy modules, users, roles, and booleans.

* **-a**: Adds a new SELinux policy module, user, role, or boolean.
  + Example: **semanage boolean -a httpd\_can\_network\_connect**
* **-d**: Deletes an existing SELinux policy module, user, role, or boolean.
  + Example: **semanage boolean -d httpd\_can\_network\_connect**
* **-l**: Lists all existing SELinux policy modules, users, roles, or booleans.
  + Example: **semanage boolean -l**
* **-m**: Modifies an existing SELinux policy module or boolean.
  + Example: **semanage boolean -m httpd\_can\_network\_connect -S on**
* **-P**: Sets the default SELinux policy.
  + Example: **semanage permissive -P policy**

1. **restorecon**: This command is used to restore the SELinux context of a file or directory to its default value.

* **-R**: Restores the SELinux context of a directory and its contents recursively.
  + Example: **restorecon -R /var/www/html**

1. **chcon**: This command is used to change the SELinux context of a file or directory.

* **-R**: Changes the SELinux context of a directory and its contents recursively.
  + Example: **chcon -R -t httpd\_sys\_content\_t /var/www/html**

1. **ls -Z**: This command displays the SELinux context of files and directories in the current directory.

* **-l**: Displays the SELinux context in a long listing format.
  + Example: **ls -Zl /var/www/html**

1. **audit2allow**: This command generates SELinux policy modules based on system audit logs.

* **-M**: Writes the generated policy module to a file.
  + Example: **audit2allow -M mypolicy < /var/log/audit/audit.log**
* **-o**: Writes the generated policy module to standard output.
  + Example: **audit2allow -o < /var/log/audit/audit.log**

1. **sealert**: This command analyzes SELinux-related log messages and provides suggestions for resolving issues.

* **-a**: Analyzes all SELinux-related log messages.
  + Example: **sealert -a /var/log/audit/audit.log**
* **-b**: Analyzes only messages related to AVC (Access Vector Cache) denials.
  + Example: **sealert -b /var/log/audit/audit.log**

continue from sealert

Sure! Here are a few more flags that you can use with the **sealert** command:

1. **sealert -l**: This command lists all the known SELinux alert messages.

* **-a**: Shows all the messages, including those that have already been seen and processed by **sealert**.
  + Example: **sealert -l -a**
* **-u**: Filters messages based on the specified user.
  + Example: **sealert -l -u apache**
* **-t**: Filters messages based on the specified type.
  + Example: **sealert -l -t mysqld**

1. **setsebool**: This command is used to modify SELinux booleans.

* **-P**: Persists the changes across reboots.
  + Example: **setsebool -P httpd\_can\_network\_connect on**
* **-N**: Does not persist the changes across reboots.
  + Example: **setsebool -N httpd\_can\_network\_connect off**

1. **ausearch**: This command searches audit logs for events that match certain criteria.

* **-m**: Searches for events based on the specified message type.
  + Example: **ausearch -m AVC**
* **-ts**: Searches for events that occurred after the specified timestamp.
  + Example: **ausearch -ts today**
* **-ui**: Searches for events performed by the specified user ID.
  + Example: **ausearch -ui 1000**

1. **auditctl**: This command is used to manage the audit subsystem.

* **-a**: Adds a new rule to the audit subsystem.
  + Example: **auditctl -a always,exit -F arch=b64 -S execve**
* **-d**: Deletes an existing rule from the audit subsystem.
  + Example: **auditctl -d always,exit -F arch=b64 -S execve**
* **-l**: Lists all the rules currently defined in the audit subsystem.
  + Example: **auditctl -l**

These are some commonly used SELinux commands and their associated flags. However, there are many more options available for each command that you can explore in the respective command's manual page or by using the **--help** option.

<https://www.whitewinterwolf.com/posts/2017/09/08/selinux-cheatsheet/>

. <https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/using_selinux/index>

**. how to allow access to file test.txt from a specific IP :SELinux**

how to allow access to file test.txt from a specific IP :SELinux

To allow access to a file "test.txt" from a specific IP address using SELinux, you can follow these steps:

1. Identify the IP address you want to allow access from. For example, let's say the IP address is "192.168.1.100".
2. Use the "semanage" command to add a new SELinux policy rule that allows read access to the file from the specified IP address. Run the following command as the root user:

**semanage port -a -t FILE\_TYPE -p PROTOCOL -s 192.168.1.100/32 FILE\_PORT**

Replace **FILE\_TYPE** with the file type of "test.txt", such as **httpd\_sys\_content\_t** for a web content file, and **PROTOCOL** with the protocol used to access the file, such as **tcp**. Replace **FILE\_PORT** with the port number used to access the file, such as **80** for HTTP or **22** for SSH.

For example, if "test.txt" is a web content file and you want to allow HTTP access from the specified IP address, you would run the following command:

**semanage port -a -t httpd\_sys\_content\_t -p tcp -s 192.168.1.100/32 80**

1. Reload the SELinux policy to apply the changes. Run the following command:

**restorecon -R -v /path/to/test.txt**

Replace **/path/to/test.txt** with the actual path to the file.

These steps should allow access to the "test.txt" file from the specified IP address while still enforcing SELinux security policies.

.

**how to allow access to test\_directory from a specific IP :SELinux**

To allow access to a directory "test\_directory" from a specific IP address using SELinux, you can follow these steps:

1. Identify the IP address you want to allow access from. For example, let's say the IP address is "192.168.1.100".
2. Use the "semanage" command to add a new SELinux policy rule that allows read access to the directory from the specified IP address. Run the following command as the root user:

**semanage permissive -a type -s system\_u -r s0 -t FILE\_TYPE -c dir -C read -p tcp -B 192.168.1.100/32**

Replace **FILE\_TYPE** with the file type of "test\_directory", such as **httpd\_sys\_content\_t** for a web content directory, and **tcp** with the protocol used to access the directory.

For example, if "test\_directory" is a web content directory and you want to allow HTTP access from the specified IP address, you would run the following command:

**semanage permissive -a httpd\_sys\_content\_t -s system\_u -r s0 -t httpd\_sys\_content\_t -c dir -C read -p tcp -B 192.168.1.100/32**

1. Reload the SELinux policy to apply the changes. Run the following command:

**restorecon -R -v /path/to/test\_directory**

Replace **/path/to/test\_directory** with the actual path to the directory.

These steps should allow access to the "test\_directory" directory from the specified IP address while still enforcing SELinux security policies.

.

**how to allow access to aspecific service like "vsftpd" from a specific IP :SELinux**

. To allow access to a specific service like "vsftpd" from a specific IP address using SELinux, you can follow these steps:

1. Identify the IP address you want to allow access from. For example, let's say the IP address is "192.168.1.100".
2. Use the "semanage" command to add a new SELinux policy rule that allows access to the vsftpd service from the specified IP address. Run the following command as the root user:

**semanage port -a -t ftpd\_port\_t -p tcp -s 192.168.1.100/32 21**

1. This command allows TCP traffic to port 21 (used by the vsftpd service) from the specified IP address. Replace **ftpd\_port\_t** with the appropriate SELinux type for your vsftpd installation.
2. Reload the SELinux policy to apply the changes. Run the following command:

**restorecon -R -v /path/to/vsftpd**

1. Replace **/path/to/vsftpd** with the actual path to the vsftpd service.

These steps should allow access to the vsftpd service from the specified IP address while still enforcing SELinux security policies.