Access Control Lists (ACLs)

1. Introduction

An Access Control List (ACL) is a set of rules used to control network traffic and enhance security by determining which users or systems are allowed or denied access to resources. ACLs are widely used in computer networks and operating systems to filter traffic, manage permissions, and protect critical resources.

On computer networks, ACLs can be installed on routers or switches, where they act as traffic filters, allowing administrators to regulate the type of traffic entering or leaving the network. Similarly, in operating systems, ACLs manage access to files, directories, and applications, ensuring that only authorized users can interact with them.

2. Usability of ACLs

ACLs are used for a variety of purposes across network environments and operating systems:

- **Traffic Filtering:** ACLs can allow or deny network traffic based on source/destination IP addresses, ports, or protocols.
- **User Management:** Filesystem ACLs manage user permissions, defining read, write, or execute access to specific files and directories.
- **Network Security:** By blocking malicious or unwanted traffic, ACLs help prevent attacks and unauthorized access.
- **Performance Enhancement:** ACLs improve network efficiency by controlling traffic at critical points, reducing unnecessary load on devices.

ACLs are commonly applied on network interfaces in two directions:

- **Inbound ACLs**: Filter traffic as it enters an interface before routing it further.
- **Outbound ACLs**: Filter traffic leaving an interface after routing it.

Placement of ACLs is strategic. Standard ACLs are typically applied close to the destination, while extended ACLs are applied near the source. This ensures maximum control and minimal impact on network performance.

3. Types of Access Control Lists (ACLs)

Access Control Lists can be categorized in multiple ways depending on what they control and how they identify traffic.

1. Based on Functionality

a) File System ACLs

These ACLs are used by operating systems to manage access to files and directories. They provide instructions that define user permissions, such as read, write, or execute, for specific system resources. Each file or directory has an associated ACL, which determines what a user can do once they access the system.

b) Networking ACLs

Networking ACLs are used to control access to computer networks. They provide rules to switches and routers, specifying which traffic is allowed or denied. These ACLs also define user permissions once they are inside the network, allowing administrators to manage traffic flow. Networking ACLs are similar to firewalls in functionality; they filter traffic based on predefined rules set by the network administrator.

2. Based on Traffic Identification

a) Standard ACLs

Block or allow traffic based solely on the source IP address. They apply the same rule to all protocols (TCP, UDP, ICMP, etc.) from the specified source. Useful for basic filtering where only the source of traffic matters.

b) Extended ACLs

Provide more granular control, filtering traffic based on source IP address, destination IP address, source and destination ports, and protocol type (TCP, UDP, ICMP, etc.). Extended ACLs are ideal for complex network environments where administrators need to differentiate traffic types and control access precisely.

4. Technical Details

4.1 ACL Components

ACL entries consist of several components that determine how traffic is filtered:

- **Sequence Number:** Identifies the entry in the ACL.
- **ACL Name/Number:** Identifies the ACL. Named ACLs use letters; numbered ACLs use numbers.
- **Statement/Action:** Specifies permit or deny rules.

- **Source & Destination:** Specifies IP addresses or ranges for filtering.
- **Network Protocol:** Defines which protocols (TCP, UDP, ICMP, IPX) are allowed or blocked.
- **Wildcard Mask**: Determines which bits of an IP address to check.
- **Log:** Some ACLs keep a log of matched entries.
- **Advanced Criteria:** Include Type of Service (ToS), Differentiated Services Code Point (DSCP), or IP precedence.

5. Applications of ACLs

5.1 Network Security

- Filtering unwanted traffic from the internet.
- Blocking malicious or suspicious network activity.
- Segregating traffic in DMZs (Demilitarized Zones) for web servers, VPNs, and DNS servers.

5.2 Operating System Access Control

- Restricting access to files and directories.
- Assigning granular privileges to different users or groups.
- Ensuring only authorized users can execute sensitive programs.

5.3 Cloud and Enterprise Networks

- Controlling traffic between on-premises networks and cloud resources.
- Automating access policies for consistent network security.
- Optimizing traffic flows for performance and security compliance.

6. Advantages of ACLs

- **Improved Network Performance:** Filtering unnecessary traffic reduces congestion and improves routing efficiency.

- **Enhanced Security:** ACLs allow administrators to permit or deny access, mitigating unauthorized access and attacks.

- **Granular Control:** Administrators can manage traffic and user permissions with precision.

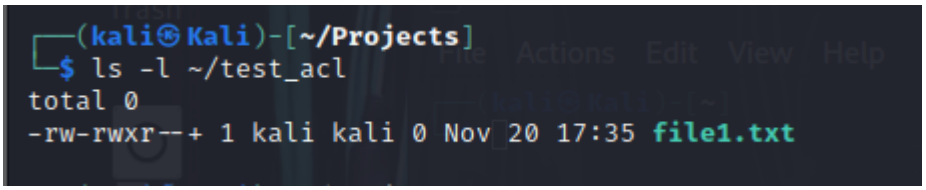
- **Scalability**: ACLs can be applied at multiple points in a network, including edge routers, DMZs, and endpoints.

7. Practical Implementation of ACLs on Kali Linux

Filesystem ACL:

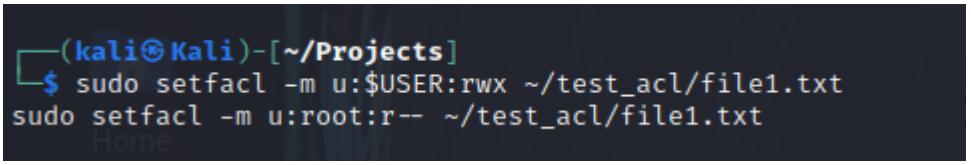
In this practical, we demonstrate how to create files, set ACL permissions, and check ACL entries on Kali Linux.

Step 1: Create Test Directory and File



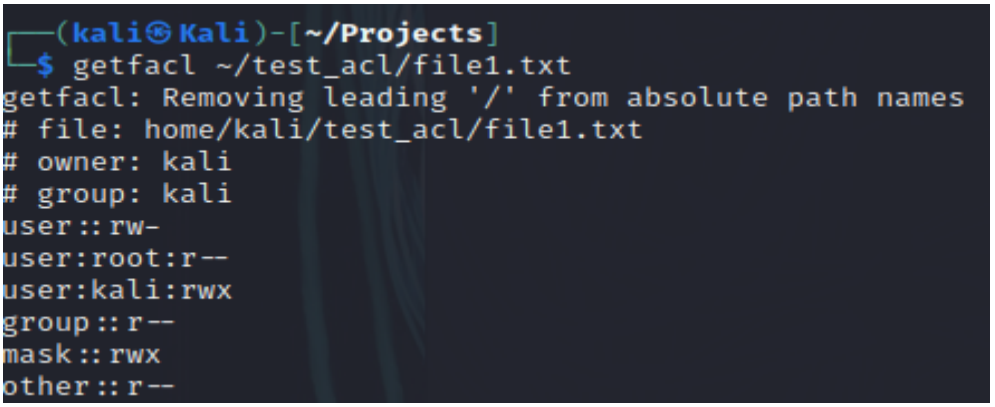
The + sign at the end indicates ACL is enabled on the file.

Step 2: Set ACL Permissions



Set ACL permissions on the file, giving `Kali` full access (`rwx`) and `root` read-only (`r--`).

Step 3: Check ACL Entries



Verify the ACL using `getfacl` to confirm that the correct permissions were applied to each user.

Networking ACL

Step 1: Check file and its permissions

```
┌──(kali㉿Kali)-[~/Projects]
└─$ sudo iptables -L -n -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy DROP 952 packets, 9188K bytes)
 pkts bytes target     prot opt in     out     source               destination
 1240 9205K DOCKER-USER  0    --  *      *       0.0.0.0/0            0.0.0.0/0
 1240 9205K DOCKER-ISOLATION-STAGE-1  0    --  *      *       0.0.0.0/0            0.0.0.0/0
    1    76 ACCEPT     0    --  *      docker0  0.0.0.0/0            0.0.0.0/0           ctstate RELATED,ESTABLISHED
    0     0 DOCKER     0    --  *      docker0  0.0.0.0/0            0.0.0.0/0
    3   228 ACCEPT     0    --  docker0 !docker0  0.0.0.0/0            0.0.0.0/0
    0     0 ACCEPT     0    --  docker0 docker0  0.0.0.0/0            0.0.0.0/0
    0     0 ACCEPT     0    --  br-internal br-internal  0.0.0.0/0            0.0.0.0/0
    0     0 ACCEPT     0    --  br-355ee7945a88 br-355ee7945a88  0.0.0.0/0            0.0.0.0/0
    0     0 ACCEPT     0    --  br-339414195aeb br-339414195aeb  0.0.0.0/0            0.0.0.0/0

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain DOCKER (1 references)
 pkts bytes target     prot opt in     out     source               destination

Chain DOCKER-ISOLATION-STAGE-1 (1 references)
 pkts bytes target     prot opt in     out     source               destination
    3   228 DOCKER-ISOLATION-STAGE-2  0    --  docker0 !docker0  0.0.0.0/0            0.0.0.0/0
    0     0 DROP       0    --  *      br-internal !10.6.6.0/24          0.0.0.0/0
  140  8179 DROP       0    --  br-internal *       0.0.0.0/0           !10.6.6.0/24
    0     0 DROP       0    --  *      br-355ee7945a88 !192.168.0.0/24       0.0.0.0/0
    3   228 DROP       0    --  br-355ee7945a88 *       0.0.0.0/0           !192.168.0.0/24
    0     0 DROP       0    --  *      br-339414195aeb !10.5.5.0/24          0.0.0.0/0
  141  8233 DROP       0    --  br-339414195aeb *       0.0.0.0/0           !10.5.5.0/24
  956 9188K RETURN     0    --  *      *       0.0.0.0/0            0.0.0.0/0

Chain DOCKER-ISOLATION-STAGE-2 (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DROP       0    --  *      docker0  0.0.0.0/0            0.0.0.0/0
    3   228 RETURN     0    --  *      *       0.0.0.0/0            0.0.0.0/0

Chain DOCKER-USER (1 references)
 pkts bytes target     prot opt in     out     source               destination
 1240 9205K RETURN     0    --  *      *       0.0.0.0/0            0.0.0.0/0
```

Shows current ACL rules.

Step 2: Block a specific IP

```
┌──(kali㉿Kali)-[~/Projects]
└─$ sudo iptables -A INPUT -s 192.168.1.10 -j DROP
```

Deny traffic from a specific IP

Step 3: Allow a specific IP

```
┌──(kali㉿Kali)-[~/Projects]
└─$ sudo iptables -A INPUT -s 192.168.1.20 -j ACCEPT
```

Permit traffic from a trusted IP

Step 4: Block a specific port (SSH, port 22)

```
┌──(kali㉿Kali)-[~/Projects]
└─$ sudo iptables -A INPUT -p tcp --dport 22 -j DROP
```

Block access to a specific port

Step 5: Allow a specific port (HTTP, port 80)

```
┌──(kali㉿Kali)-[~/Projects]
└─$ sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

Permit web traffic on HTTP port.

Step 6: Delete a rule (Undo)

```
┌──(kali㉿Kali)-[~/Projects]
└─$ sudo iptables -L INPUT -n --line-numbers
Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination
1    ACCEPT     0    --  192.168.1.20         0.0.0.0/0
2    ACCEPT     6    --  0.0.0.0/0            0.0.0.0/0           tcp dpt:80
```

```
┌──(kali㉿Kali)-[~/Projects]
└─$ sudo iptables -D INPUT 1

┌──(kali㉿Kali)-[~/Projects]
└─$ sudo iptables -L INPUT -n --line-numbers
Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination
1    ACCEPT     6    --  0.0.0.0/0            0.0.0.0/0           tcp dpt:80

┌──(kali㉿Kali)-[~/Projects]
└─$ sudo iptables -D INPUT 1

┌──(kali㉿Kali)-[~/Projects]
└─$ sudo iptables -L INPUT -n --line-numbers
Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination
```

Deleting previously added rules.

7. Conclusion

Access Control Lists are a fundamental tool for network and system security, providing a structured method to regulate traffic and user access. By combining standard and extended ACLs, and using numbered or named lists, administrators can implement precise, flexible, and efficient access policies. ACLs not only improve security but also enhance network performance, making them an essential component of modern IT infrastructure.