Report: Implementation of Access Control Measures

1. Access Control List (ACL) Configuration

- **Objective**: Restrict access to a server such that only a specific range of IPs can connect via SSH, while all other traffic is denied.
- **Scenario**:
    - Allow SSH access only from the subnet `192.168.1.0/24`.
    - Deny all other traffic by default.

ACL Configuration (example using a Cisco Router):

```
ip access-list extended SSH_ACCESS permit tcp 192.168.1.0 0.0.0.255 any eq 22 deny
ip any any
```

Implementation Steps:

1. Access the router configuration terminal.
2. Define the ACL (`SSH_ACCESS`) and specify the rules:
    - **Permit** SSH (TCP on port 22) from `192.168.1.0/24`.
    - **Deny** all other traffic.
3. Apply the ACL to the inbound traffic of the router interface:
4. `interface GigabitEthernet0/0 ip access-group SSH_ACCESS in`

---

2. Access Control Model

- **Model Used**: **Discretionary Access Control (DAC)**
- **Objective**: Allow the owner of a file or resource to define access permissions for other users.

Example Implementation:

- **Scenario**: A file named `project_data.txt` should only be accessible by the owner (`alice`) and a specific user (`bob`).
- **Steps** (Linux File Permissions):
    1. Set ownership of the file:
    2. `chown alice:users project_data.txt`
    3. Configure permissions to allow only the owner and a specific group (`users`) to read/write:

4. `chmod 640 project_data.txt`
    - Owner (`alice`): Read and Write.
    - Group (`users`): Read only.
    - Others: No access.
5. Add `bob` to the `users` group to grant him access:
6. `usermod -aG users bob`

---

3. User Access Level

- **Objective**: Implement role-based access control (RBAC) by assigning distinct access levels for users based on roles.

Example Implementation:

- **Roles**:
    - **Admin**: Full control over all resources.
    - **Manager**: Access to reports and configuration settings but no system-wide changes.
    - **Employee**: Read-only access to reports.

Implementation Steps:

1. Define user roles in a system (e.g., Active Directory or Linux system).
    - **Linux Example**: Use groups to define roles.
    - `groupadd admins groupadd managers groupadd employees`
2. Assign users to groups based on their roles:
3. `usermod -aG admins admin_user usermod -aG managers manager_user usermod -aG employees employee_user`
4. Restrict file access based on roles:
    - Admins have full access:
    - `chown root:admins /etc/important_config chmod 770 /etc/important_config`
    - Managers have read and execute access to specific directories:
    - `chown root:managers /home/manager_reports chmod 750 /home/manager_reports`
    - Employees have read-only access:

- `chown root:employees /home/public_data chmod 740 /home/public_data`

Example Scenario

- **Scenario**:
  - Alice (Admin) configures a sensitive file for restricted access.
  - Bob (Manager) reads the project report file but cannot modify it.
  - Charlie (Employee) has read-only access to public resources.

Test Case:

- Bob attempts to edit `project_data.txt` but is denied due to permission settings (`chmod 640`).
- Alice successfully modifies the file since she is the owner.
- Charlie cannot access `project_data.txt` because he is not in the allowed group.
