Here's how to set up authorization on MongoDB securely on Ubuntu in accordance with the CIS benchmarks:

1. Enable Authorization:

- Edit the MongoDB configuration file (/etc/mongodb.conf) with root privileges.
- Locate the security section or add one if it doesn't exist.
- Set the authorization parameter to enabled. This enforces user access control through roles and permissions.

YAML

```yaml
security:
  authorization: enabled
```

2. Create Roles:

- Use the db.createRole command in the admin database to define roles with specific permissions. The CIS benchmarks might recommend pre-defined roles with varying levels of access (read-only, read-write, etc.) or custom roles depending on your needs.

Example Role Creation:

```
mongo
use admin
db.createRole(
  {
    role: "readWriteUser",
    privileges: [
      { resource: { db: "myDatabase", collection: "users" }, actions:
["find", "update"] },
      { resource: { db: "anotherDatabase", collection: "logs" },
actions: ["find"] }
    ],
    roles: []
  }
)
```

- This example creates a "readWriteUser" role with:
  - Read and update access to the "users" collection in the "myDatabase" database.
  - Read-only access to the "logs" collection in the "anotherDatabase" database.

3. Create Users and Assign Roles:

- Use the db.createUser command in the admin database to create user accounts.
- Assign the previously defined roles to the users.

Example User Creation:

```
db.createUser(
  {
    user: "username",
    pwd: passwordHash("strong_password"),
    roles: [ "readWriteUser" ]
  }
)
```

- This creates a user named "username" with the "readWriteUser" role assigned.

4. Access Control Documents (ACDs) (Optional):

- The CIS benchmarks might recommend using Access Control Documents (ACDs) for more granular control within collections.
- ACDs allow specifying permissions at the collection level, overriding role-based permissions for specific users or operations.

5. Secure Password Management:

- Enforce strong password policies with minimum length and complexity requirements.
- Consider using a password manager for secure storage and retrieval of user credentials.

6. Best Practices:

- Use the principle of least privilege. Grant users only the minimum permissions required for their tasks.
- Regularly review and update user roles and permissions to reflect changes in user responsibilities.
- Consider auditing user activity to identify suspicious access patterns.

Additional Resources:

- CIS MongoDB Security Benchmarks (reference security guides that mention them): While not available directly online, you can find references and explanations in security guides that reference CIS benchmarks. Search for "CIS MongoDB Security Benchmarks Ubuntu".
- MongoDB Documentation on Authorization: https://www.mongodb.com/docs/manual/core/authentication/
- MongoDB Documentation on Roles:

https://www.mongodb.com/docs/manual/tutorial/manage-users-and-roles/
- MongoDB Documentation on Access Control: https://www.mongodb.com/docs/manual/tutorial/enable-authentication/

Remember:

- Consult the specific CIS MongoDB benchmarks for your Ubuntu version for detailed recommendations on authorization configuration and best practices.