Role-Based Access Control (RBAC)

Project Introduction

This project simulates a real-world **Role-Based Access Control (RBAC)** setup for a company named **BagmatiCyber Ltd**.
As a junior system administrator, my responsibility was to configure Linux users, groups, file permissions, sudo policies, and automated backups to ensure secure and controlled access for different departments: Development, Audit & Compliance, and Operations.

The goal of this project was to apply Linux system administration concepts such as user management, ownership, permissions, SetGID, sudo restrictions, cron jobs, and access control in a practical environment.

Objectives

- Implement RBAC
- Secure access to files
- Configure sudo, cron, aliases

Implementation

- User & Group Setup
- Directory & File Setup
- Permissions & Ownership
- Sudo Configuration
- Testing
- Cron & Aliases

What I Learned

- Linux permissions
- SetGID
- RBAC
- Security best practices

Part 1:User and Group Setup

User Creation and Group Assignment

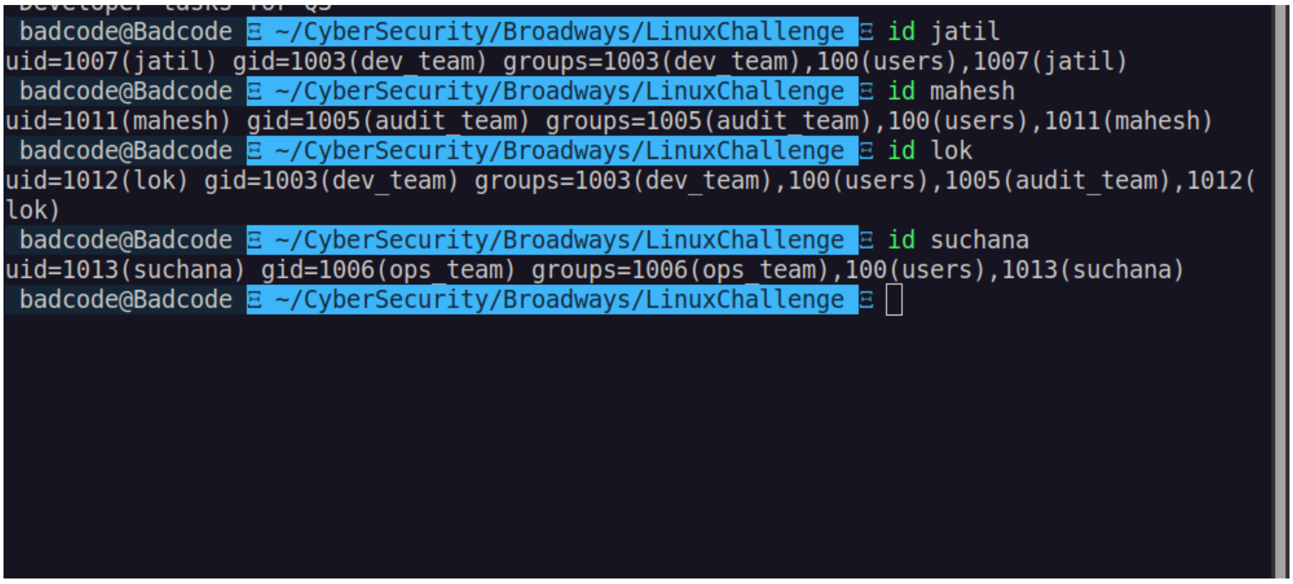
This task involved creating users and assigning them to their respective departmental groups to implement **Role-Based Access Control (RBAC)**.

Each user was assigned a **primary group** based on their department to control file ownership, and a **secondary group** to allow additional access where required:

- **Jatil** was assigned to the `dev_team` as the primary group to manage development-related files.
- **Mahesh** was assigned to the `audit_team` to handle audit and compliance tasks.
- **Lok** was assigned to both `dev_team` and `audit_team` because his role required access to both development and audit resources.
- **Suchana** was assigned to the `ops_team` to manage operational system tasks.

The **primary group** determines the default group ownership of files created by the user, while **secondary groups** provide extra access permissions without changing file ownership.

After assigning the groups, the `id` and `groups` commands were used to **verify group membership** for each user. This ensured that all users were correctly assigned to their intended departments and access levels.



```
 badcode@Badcode  ~/CyberSecurity/Broadways/LinuxChallenge  id jatil
uid=1007(jatil) gid=1003(dev_team) groups=1003(dev_team),100(users),1007(jatil)
 badcode@Badcode  ~/CyberSecurity/Broadways/LinuxChallenge  id mahesh
uid=1011(mahesh) gid=1005(audit_team) groups=1005(audit_team),100(users),1011(mahesh)
 badcode@Badcode  ~/CyberSecurity/Broadways/LinuxChallenge  id lok
uid=1012(lok) gid=1003(dev_team) groups=1003(dev_team),100(users),1005(audit_team),1012(
lok)
 badcode@Badcode  ~/CyberSecurity/Broadways/LinuxChallenge  id suchana
uid=1013(suchana) gid=1006(ops_team) groups=1006(ops_team),100(users),1013(suchana)
 badcode@Badcode  ~/CyberSecurity/Broadways/LinuxChallenge 
```

Part 2: Directory and File Setup

In this step, the required directory structure was created inside:

`/home/badcode/CyberSecurity/Broadways/LinuxChallenge` to organize departmental resources securely for BagmatiCyber Ltd.

Two main directories were created:

- **projects** – for development-related files
- **logs** – for audit and compliance records

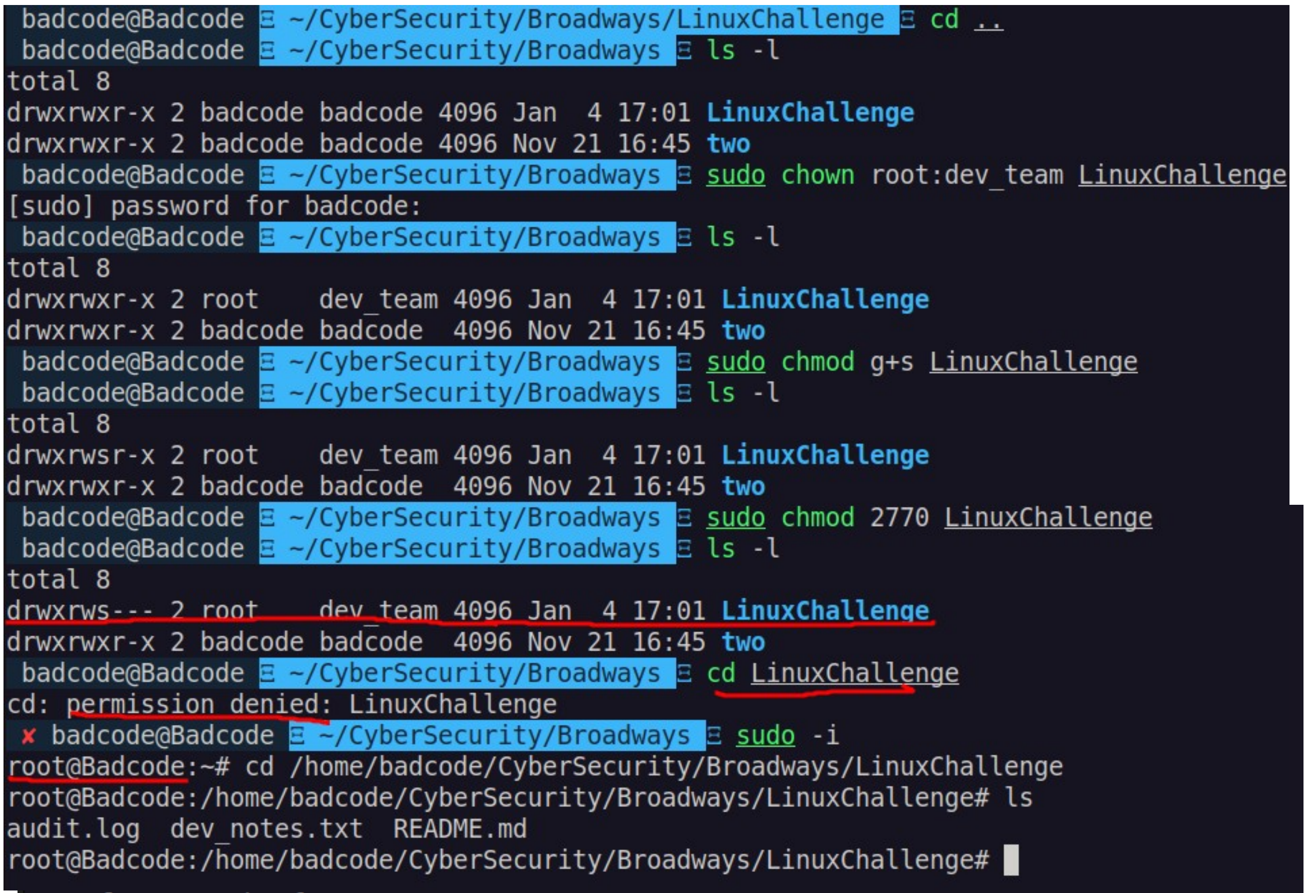
These directories help separate sensitive data based on department roles, making access control easier to manage.

Next, the following files were created with specific content:

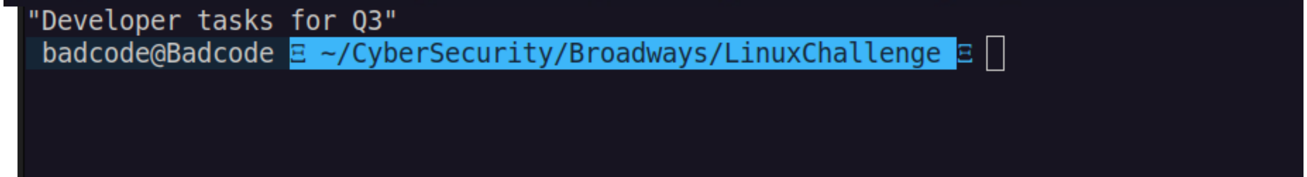
- `dev_notes.txt` inside the **projects** directory, containing development task information.
- `audit.log` inside the **logs** directory, containing audit trail records.
- `README.md` in the main **LinuxChallenge** directory, containing the company's access control policy.

This structure ensures that:

- Development, audit, and policy files are stored in appropriate locations.
- Permission rules can be applied accurately for each department.
- Sensitive data is logically separated to improve security and management.



```
 badcode@Badcode  ~/CyberSecurity/Broadways/LinuxChallenge  cd ..
 badcode@Badcode  ~/CyberSecurity/Broadways  ls -l
total 8
drwxrwxr-x 2 badcode badcode 4096 Jan  4 17:01 LinuxChallenge
drwxrwxr-x 2 badcode badcode 4096 Nov 21 16:45 two
 badcode@Badcode  ~/CyberSecurity/Broadways  sudo chown root:dev_team LinuxChallenge
[sudo] password for badcode:
 badcode@Badcode  ~/CyberSecurity/Broadways  ls -l
total 8
drwxrwxr-x 2 root    dev_team 4096 Jan  4 17:01 LinuxChallenge
drwxrwxr-x 2 badcode badcode  4096 Nov 21 16:45 two
 badcode@Badcode  ~/CyberSecurity/Broadways  sudo chmod g+s LinuxChallenge
 badcode@Badcode  ~/CyberSecurity/Broadways  ls -l
total 8
drwxrwsr-x 2 root    dev_team 4096 Jan  4 17:01 LinuxChallenge
drwxrwxr-x 2 badcode badcode  4096 Nov 21 16:45 two
 badcode@Badcode  ~/CyberSecurity/Broadways  sudo chmod 2770 LinuxChallenge
 badcode@Badcode  ~/CyberSecurity/Broadways  ls -l
total 8
drwxrws--- 2 root    dev_team 4096 Jan  4 17:01 LinuxChallenge
drwxrwxr-x 2 badcode badcode  4096 Nov 21 16:45 two
 badcode@Badcode  ~/CyberSecurity/Broadways  cd LinuxChallenge
cd: permission denied: LinuxChallenge
 ✘ badcode@Badcode  ~/CyberSecurity/Broadways  sudo -i
root@Badcode:~# cd /home/badcode/CyberSecurity/Broadways/LinuxChallenge
root@Badcode:/home/badcode/CyberSecurity/Broadways/LinuxChallenge# ls
audit.log  dev_notes.txt  README.md
root@Badcode:/home/badcode/CyberSecurity/Broadways/LinuxChallenge# 
```



```
 "Developer tasks for Q3"
  badcode@Badcode  ~/CyberSecurity/Broadways/LinuxChallenge  
```

Part 3: Ownership and Permission Configuration

In this step, file ownership was configured to ensure that only the appropriate users and departments could manage specific files.

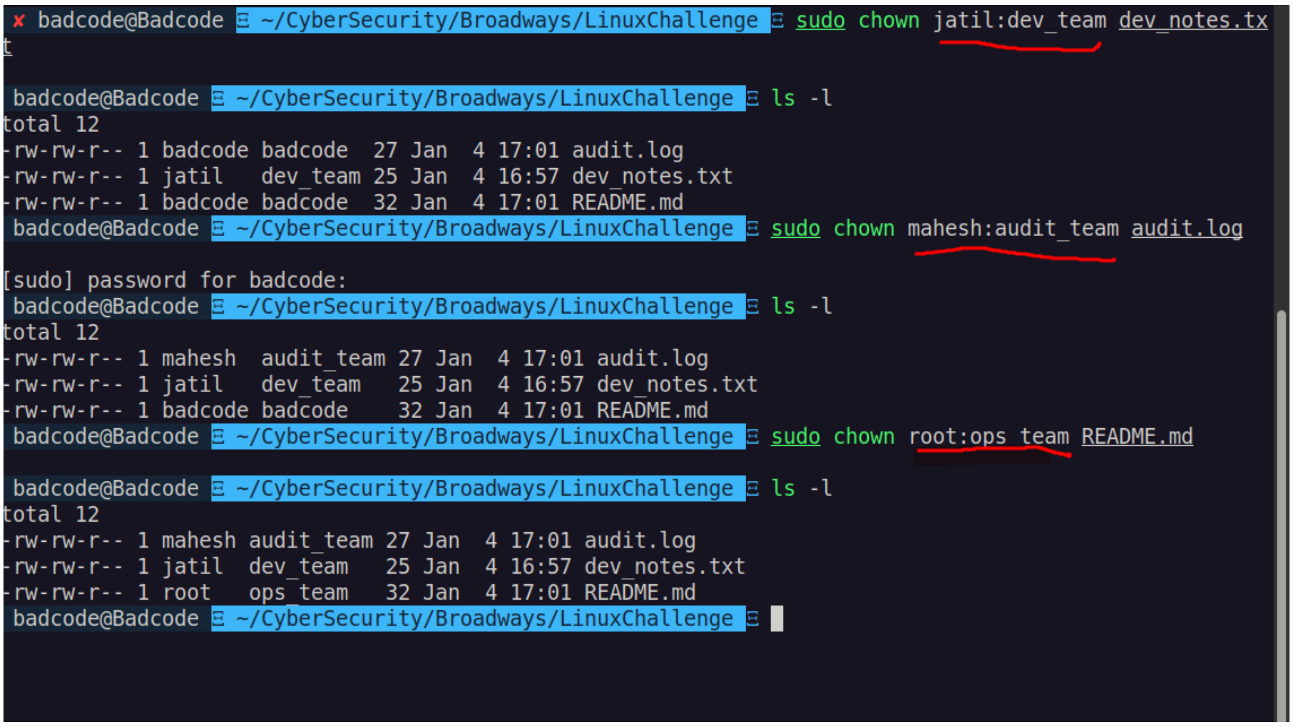
The following ownership rules were applied:

- **dev_notes.txt** was owned by **Jatil** and the group **dev_team**, allowing the development team to manage development-related tasks.
- **audit.log** was owned by **Mahesh** and the group **audit_team**, ensuring audit records are controlled by the audit department.
- **README.md** was owned by **root** and the group **ops_team**, since it contains important system access policies.

By assigning correct **user ownership** and **group ownership**, access control was enforced based on departmental roles. This ensures that:

- Only authorized users can modify sensitive files.
- Each department is responsible for its own data.
- System policies remain protected under administrative control.

This setup is a key part of implementing **Role-Based Access Control (RBAC)** in the Linux environment.



```
✘ badcode@Badcode  ~/CyberSecurity/Broadways/LinuxChallenge  sudo chown jatil:dev_team dev_notes.tx
t

 badcode@Badcode  ~/CyberSecurity/Broadways/LinuxChallenge  ls -l
total 12
-rw-rw-r-- 1 badcode badcode  27 Jan  4 17:01 audit.log
-rw-rw-r-- 1 jatil   dev_team 25 Jan  4 16:57 dev_notes.txt
-rw-rw-r-- 1 badcode badcode  32 Jan  4 17:01 README.md
 badcode@Badcode  ~/CyberSecurity/Broadways/LinuxChallenge  sudo chown mahesh:audit_team audit.log

[sudo] password for badcode:
 badcode@Badcode  ~/CyberSecurity/Broadways/LinuxChallenge  ls -l
total 12
-rw-rw-r-- 1 mahesh  audit_team 27 Jan  4 17:01 audit.log
-rw-rw-r-- 1 jatil   dev_team   25 Jan  4 16:57 dev_notes.txt
-rw-rw-r-- 1 badcode badcode    32 Jan  4 17:01 README.md
 badcode@Badcode  ~/CyberSecurity/Broadways/LinuxChallenge  sudo chown root:ops_team README.md

 badcode@Badcode  ~/CyberSecurity/Broadways/LinuxChallenge  ls -l
total 12
-rw-rw-r-- 1 mahesh audit_team 27 Jan  4 17:01 audit.log
-rw-rw-r-- 1 jatil  dev_team   25 Jan  4 16:57 dev_notes.txt
-rw-rw-r-- 1 root   ops_team   32 Jan  4 17:01 README.md
 badcode@Badcode  ~/CyberSecurity/Broadways/LinuxChallenge 
```

Directory Permissions and SetGID Configuration

In this step, strict access control was applied to the **projects** and **logs** directories to protect sensitive departmental data.

The **projects** directory was configured so that only the **root user** and members of the **dev_team** can access it.
The **logs** directory was configured so that only the **root user** and members of the **audit_team** can access it.

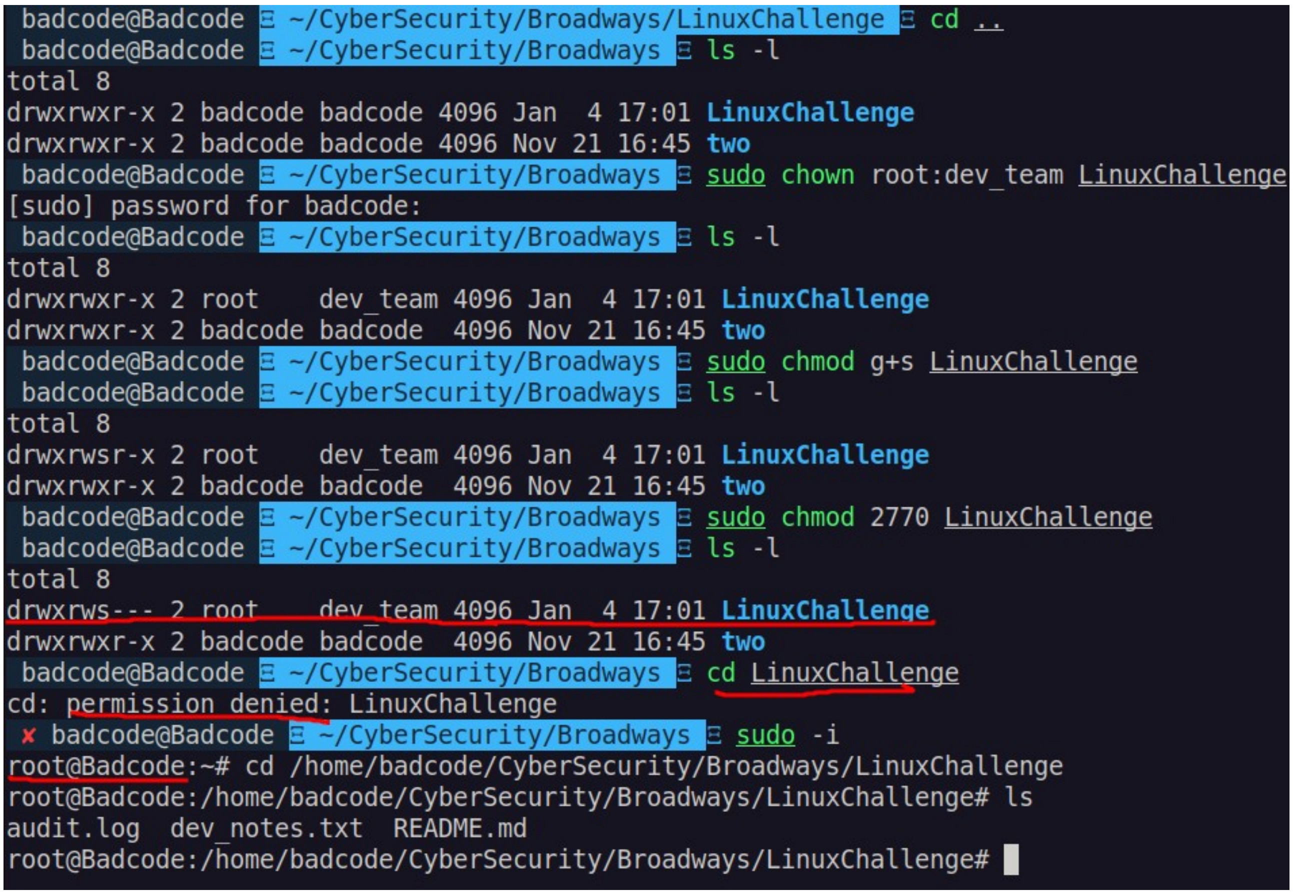
The **SetGID (Set Group ID)** permission was enabled on both directories.
This ensures that any new files created inside these directories automatically inherit the group ownership of the directory.
This is important for maintaining consistent access control within each department.

For the **README.md** file, special permissions were applied:

- The file is **readable by all users**, so everyone can view the access policy.
- **No user is allowed to write or execute** the file, including the owner.

This protects the policy document from being modified or misused.

Overall, these permission settings enforce **department-based access**, prevent unauthorized changes, and maintain a secure RBAC environment.

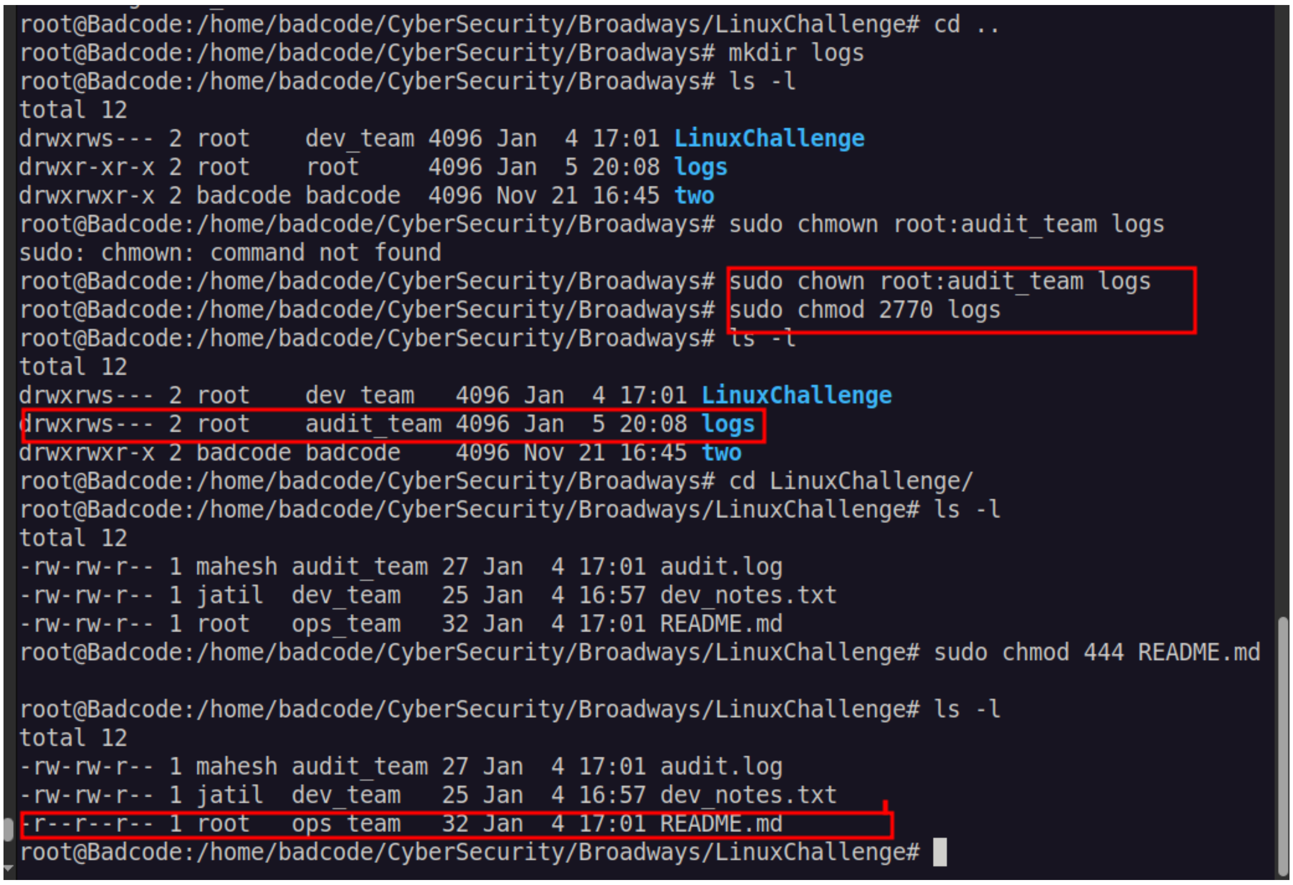


```
 badcode@Badcode  ~/CyberSecurity/Broadways/LinuxChallenge  cd ..
 badcode@Badcode  ~/CyberSecurity/Broadways  ls -l
total 8
drwxrwxr-x 2 badcode badcode 4096 Jan  4 17:01 LinuxChallenge
drwxrwxr-x 2 badcode badcode 4096 Nov 21 16:45 two
 badcode@Badcode  ~/CyberSecurity/Broadways  sudo chown root:dev_team LinuxChallenge
[sudo] password for badcode:
 badcode@Badcode  ~/CyberSecurity/Broadways  ls -l
total 8
drwxrwxr-x 2 root    dev_team 4096 Jan  4 17:01 LinuxChallenge
drwxrwxr-x 2 badcode badcode  4096 Nov 21 16:45 two
 badcode@Badcode  ~/CyberSecurity/Broadways  sudo chmod g+s LinuxChallenge
 badcode@Badcode  ~/CyberSecurity/Broadways  ls -l
total 8
drwxrwsr-x 2 root    dev_team 4096 Jan  4 17:01 LinuxChallenge
drwxrwxr-x 2 badcode badcode  4096 Nov 21 16:45 two
 badcode@Badcode  ~/CyberSecurity/Broadways  sudo chmod 2770 LinuxChallenge
 badcode@Badcode  ~/CyberSecurity/Broadways  ls -l
total 8
drwxrws--- 2 root    dev_team 4096 Jan  4 17:01 LinuxChallenge
drwxrwxr-x 2 badcode badcode  4096 Nov 21 16:45 two
 badcode@Badcode  ~/CyberSecurity/Broadways  cd LinuxChallenge
cd: permission denied: LinuxChallenge
 ✘ badcode@Badcode  ~/CyberSecurity/Broadways  sudo -i
root@Badcode:~# cd /home/badcode/CyberSecurity/Broadways/LinuxChallenge
root@Badcode:/home/badcode/CyberSecurity/Broadways/LinuxChallenge# ls
audit.log  dev_notes.txt  README.md
root@Badcode:/home/badcode/CyberSecurity/Broadways/LinuxChallenge# 
```

In this step, access controls were applied to the `-logs` directory and `README.md` file to ensure security and proper access management.

- **`-logs` directory:** Access was restricted to the owner and members of the `audit_team`. The SetGID (Set Group ID) permission was enabled on the directory, ensuring that any new files or subdirectories created inside automatically inherit the group ownership. This maintains consistent group-based access control and prevents unauthorized access to sensitive logs.
- **`README.md` file:** The file was made readable by all users so that everyone can view its contents. However, no user—including the owner—can write to or execute the file. This ensures the file remains protected from modification or misuse.

Overall, these permission settings enforce controlled access, preserve data integrity, and maintain a secure environment for team-based collaboration.



```
root@Badcode:/home/badcode/CyberSecurity/Broadways/LinuxChallenge# cd ..
root@Badcode:/home/badcode/CyberSecurity/Broadways# mkdir logs
root@Badcode:/home/badcode/CyberSecurity/Broadways# ls -l
total 12
drwxrws--- 2 root    dev_team 4096 Jan  4 17:01 LinuxChallenge
drwxr-xr-x 2 root    root     4096 Jan  5 20:08 logs
drwxrwxr-x 2 badcode badcode  4096 Nov 21 16:45 two
root@Badcode:/home/badcode/CyberSecurity/Broadways# sudo chmown root:audit_team logs
sudo: chmown: command not found
root@Badcode:/home/badcode/CyberSecurity/Broadways# sudo chown root:audit_team logs
root@Badcode:/home/badcode/CyberSecurity/Broadways# sudo chmod 2770 logs
root@Badcode:/home/badcode/CyberSecurity/Broadways# ls -l
total 12
drwxrws--- 2 root    dev_team   4096 Jan  4 17:01 LinuxChallenge
drwxrws--- 2 root    audit_team 4096 Jan  5 20:08 logs
drwxrwxr-x 2 badcode badcode    4096 Nov 21 16:45 two
root@Badcode:/home/badcode/CyberSecurity/Broadways# cd LinuxChallenge/
root@Badcode:/home/badcode/CyberSecurity/Broadways/LinuxChallenge# ls -l
total 12
-rw-rw-r-- 1 mahesh audit_team 27 Jan  4 17:01 audit.log
-rw-rw-r-- 1 jatil  dev_team   25 Jan  4 16:57 dev_notes.txt
-rw-rw-r-- 1 root   ops_team   32 Jan  4 17:01 README.md
root@Badcode:/home/badcode/CyberSecurity/Broadways/LinuxChallenge# sudo chmod 444 README.md

root@Badcode:/home/badcode/CyberSecurity/Broadways/LinuxChallenge# ls -l
total 12
-rw-rw-r-- 1 mahesh audit_team 27 Jan  4 17:01 audit.log
-rw-rw-r-- 1 jatil  dev_team   25 Jan  4 16:57 dev_notes.txt
-r--r--r-- 1 root   ops_team   32 Jan  4 17:01 README.md
root@Badcode:/home/badcode/CyberSecurity/Broadways/LinuxChallenge# 
```

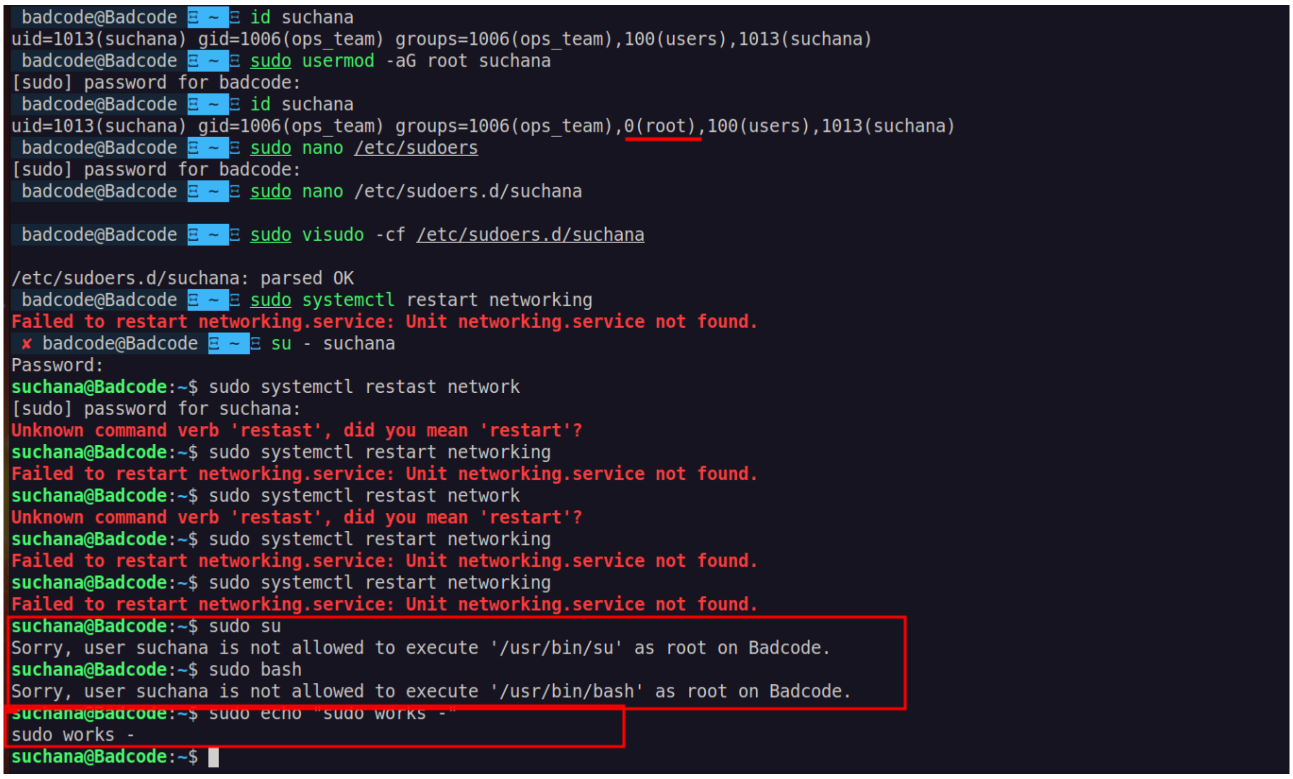
Part 4: sudo Configuration

In this step, user `Suchana` was configured with specific sudo privileges to allow controlled administrative access. She was added to the sudo group to grant her the ability to run commands with elevated privileges.

Special sudoers rules were applied to enforce security policies:

- Suchana was allowed to restart the networking service without being prompted for a password. This enables necessary administrative tasks to be performed efficiently while maintaining security.
- At the same time, she was explicitly denied the ability to use `sudo su` or `sudo bash`. This prevents her from gaining unrestricted root access and ensures that administrative privileges are used only for authorized tasks.

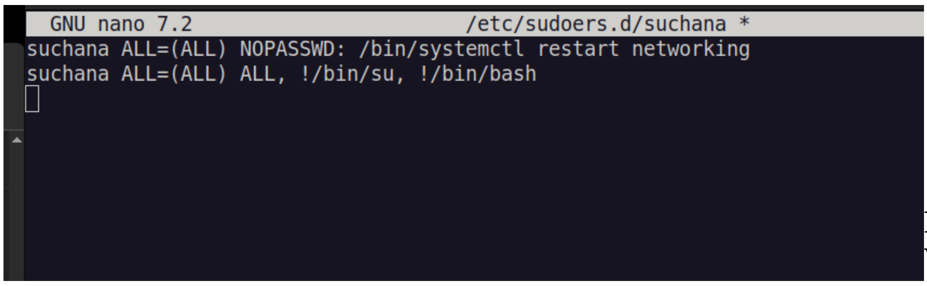
The configuration was validated by attempting the allowed and restricted commands. This approach provides controlled privilege escalation, enforcing the principle of least privilege while maintaining operational flexibility.



```
 badcode@Badcode  ~  id suchana
uid=1013(suchana) gid=1006(ops_team) groups=1006(ops_team),100(users),1013(suchana)
 badcode@Badcode  ~  sudo usermod -aG root suchana
[sudo] password for badcode:
 badcode@Badcode  ~  id suchana
uid=1013(suchana) gid=1006(ops_team) groups=1006(ops_team),0(root),100(users),1013(suchana)
 badcode@Badcode  ~  sudo nano /etc/sudoers
[sudo] password for badcode:
 badcode@Badcode  ~  sudo nano /etc/sudoers.d/suchana

 badcode@Badcode  ~  sudo visudo -cf /etc/sudoers.d/suchana

/etc/sudoers.d/suchana: parsed OK
 badcode@Badcode  ~  sudo systemctl restart networking
Failed to restart networking.service: Unit networking.service not found.
 ✘ badcode@Badcode  ~  su - suchana
Password:
suchana@Badcode:~$ sudo systemctl restast network
[sudo] password for suchana:
Unknown command verb 'restast', did you mean 'restart'?
suchana@Badcode:~$ sudo systemctl restart networking
Failed to restart networking.service: Unit networking.service not found.
suchana@Badcode:~$ sudo systemctl restast network
Unknown command verb 'restast', did you mean 'restart'?
suchana@Badcode:~$ sudo systemctl restart networking
Failed to restart networking.service: Unit networking.service not found.
suchana@Badcode:~$ sudo systemctl restart networking
Failed to restart networking.service: Unit networking.service not found.
suchana@Badcode:~$ sudo su
Sorry, user suchana is not allowed to execute '/usr/bin/su' as root on Badcode.
suchana@Badcode:~$ sudo bash
Sorry, user suchana is not allowed to execute '/usr/bin/bash' as root on Badcode.
suchana@Badcode:~$ sudo echo "sudo works -"
sudo works -
suchana@Badcode:~$
```

code inside /etc/sudoers.d/suchana

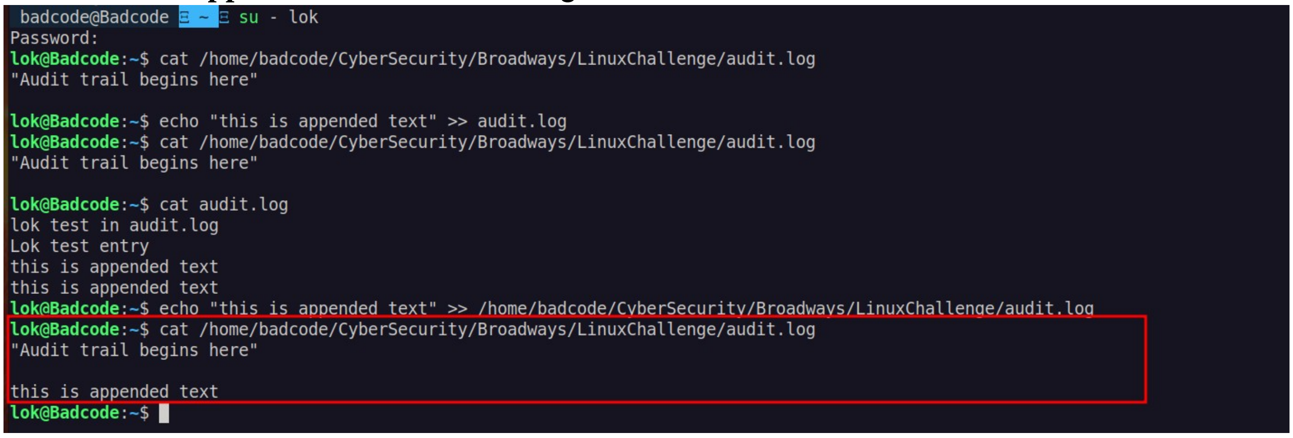
first line make no password required and 2nd code make suchana unable to use 'sudo su' and 'sudo bash'



```
  GNU nano 7.2                    /etc/sudoers.d/suchana *
suchana ALL=(ALL) NOPASSWD: /bin/systemctl restart networking
suchana ALL=(ALL) ALL, !/bin/su, !/bin/bash
```

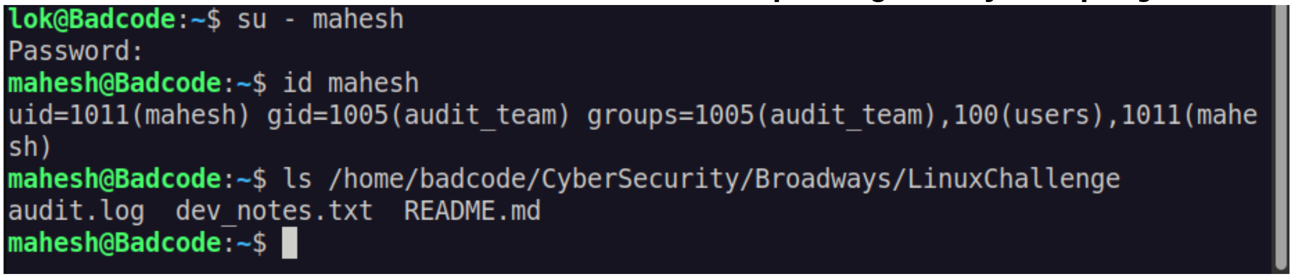
Part 5: Testing & Validation

-Lok tries to append data to audit.log



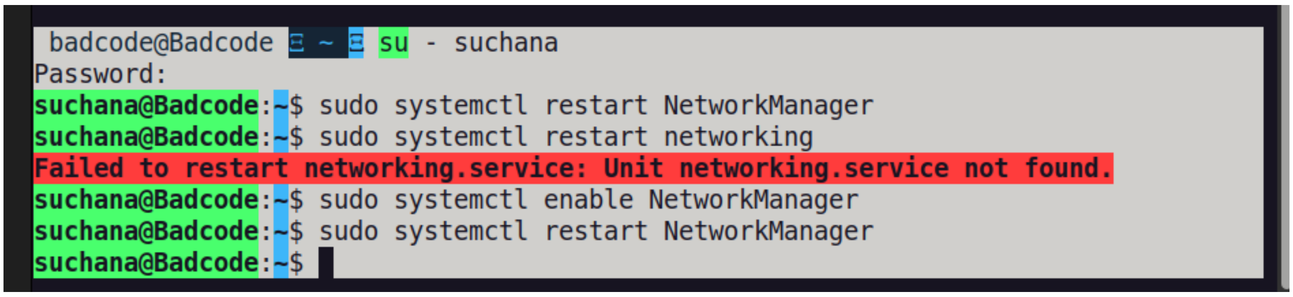
```
 badcode@Badcode  ~  su - lok
Password:
lok@Badcode:~$ cat /home/badcode/CyberSecurity/Broadways/LinuxChallenge/audit.log
"Audit trail begins here"

lok@Badcode:~$ echo "this is appended text" >> audit.log
lok@Badcode:~$ cat /home/badcode/CyberSecurity/Broadways/LinuxChallenge/audit.log
"Audit trail begins here"

lok@Badcode:~$ cat audit.log
lok test in audit.log
Lok test entry
this is appended text
this is appended text
lok@Badcode:~$ echo "this is appended text" >> /home/badcode/CyberSecurity/Broadways/LinuxChallenge/audit.log
lok@Badcode:~$ cat /home/badcode/CyberSecurity/Broadways/LinuxChallenge/audit.log
"Audit trail begins here"

this is appended text
lok@Badcode:~$
```

-Mahesh tries to list the contents of /opt/Bagmaticyber/projects



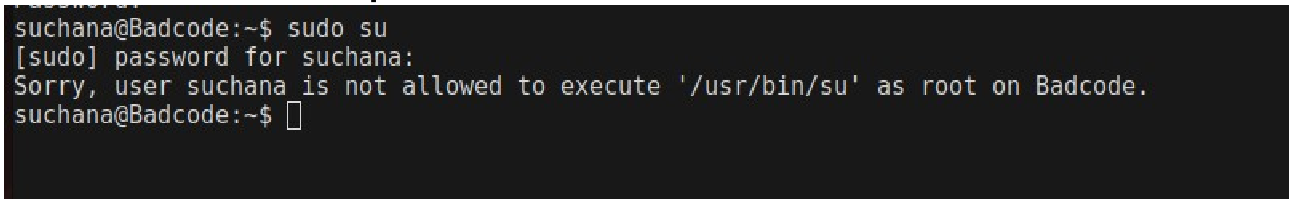
```
lok@Badcode:~$ su - mahesh
Password:
mahesh@Badcode:~$ id mahesh
uid=1011(mahesh) gid=1005(audit_team) groups=1005(audit_team),100(users),1011(mahe
sh)
mahesh@Badcode:~$ ls /home/badcode/CyberSecurity/Broadways/LinuxChallenge
audit.log  dev_notes.txt  README.md
mahesh@Badcode:~$ 
```

-Suchana restarts the network service



```
 badcode@Badcode  ~  su - suchana
Password:
suchana@Badcode:~$ sudo systemctl restart NetworkManager
suchana@Badcode:~$ sudo systemctl restart networking
Failed to restart networking.service: Unit networking.service not found.
suchana@Badcode:~$ sudo systemctl enable NetworkManager
suchana@Badcode:~$ sudo systemctl restart NetworkManager
suchana@Badcode:~$ 
```

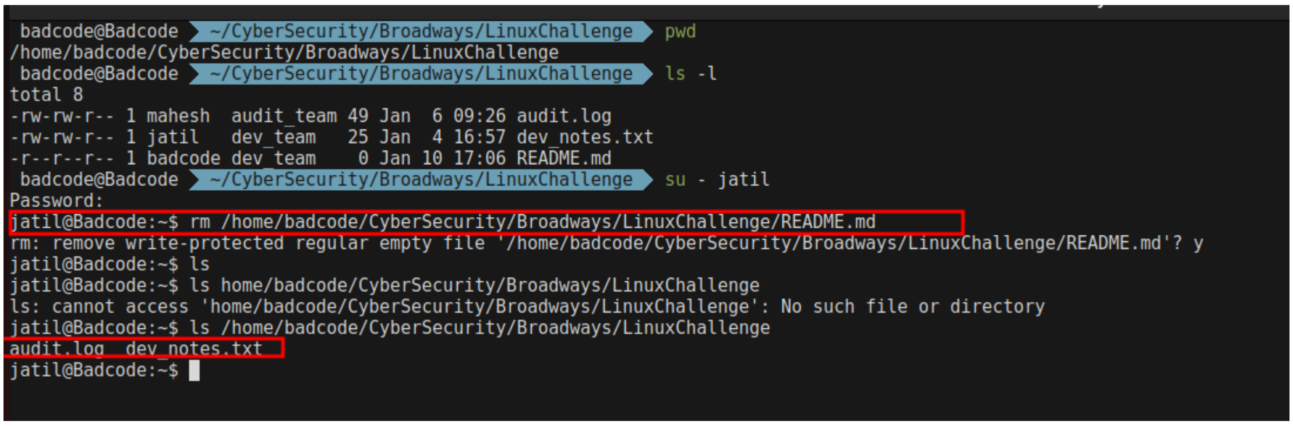
-Suchana tries to open a root shell



```
suchana@Badcode:~$ sudo su
[sudo] password for suchana:
Sorry, user suchana is not allowed to execute '/usr/bin/su' as root on Badcode.
suchana@Badcode:~$ 
```

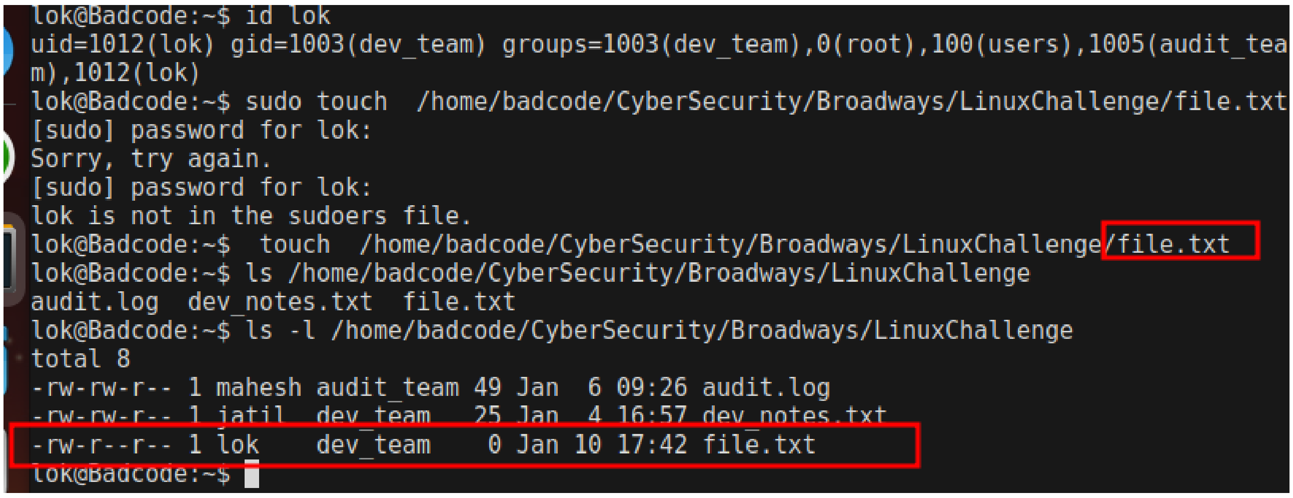
-Jatil attempts to delete README.md

The **directory /home/badcode/CyberSecurity/Broadways/LinuxChallenge has write permission** for his group (dev_team).Jatil is a member of that group.so, it can remove it.



```
 badcode@Badcode  ~/CyberSecurity/Broadways/LinuxChallenge  pwd
/home/badcode/CyberSecurity/Broadways/LinuxChallenge
 badcode@Badcode  ~/CyberSecurity/Broadways/LinuxChallenge  ls -l
total 8
-rw-rw-r-- 1 mahesh  audit_team 49 Jan  6 09:26 audit.log
-rw-rw-r-- 1 jatil   dev_team   25 Jan  4 16:57 dev_notes.txt
-r--r--r-- 1 badcode dev_team    0 Jan 10 17:06 README.md
 badcode@Badcode  ~/CyberSecurity/Broadways/LinuxChallenge  su - jatil
Password:
jatil@Badcode:~$ rm /home/badcode/CyberSecurity/Broadways/LinuxChallenge/README.md
rm: remove write-protected regular empty file '/home/badcode/CyberSecurity/Broadways/LinuxChallenge/README.md'? y
jatil@Badcode:~$ ls
jatil@Badcode:~$ ls home/badcode/CyberSecurity/Broadways/LinuxChallenge
ls: cannot access 'home/badcode/CyberSecurity/Broadways/LinuxChallenge': No such file or directory
jatil@Badcode:~$ ls /home/badcode/CyberSecurity/Broadways/LinuxChallenge
audit.log  dev_notes.txt
jatil@Badcode:~$ 
```

-Lok creates a new file inside /opt/Bagmaticyber/logs

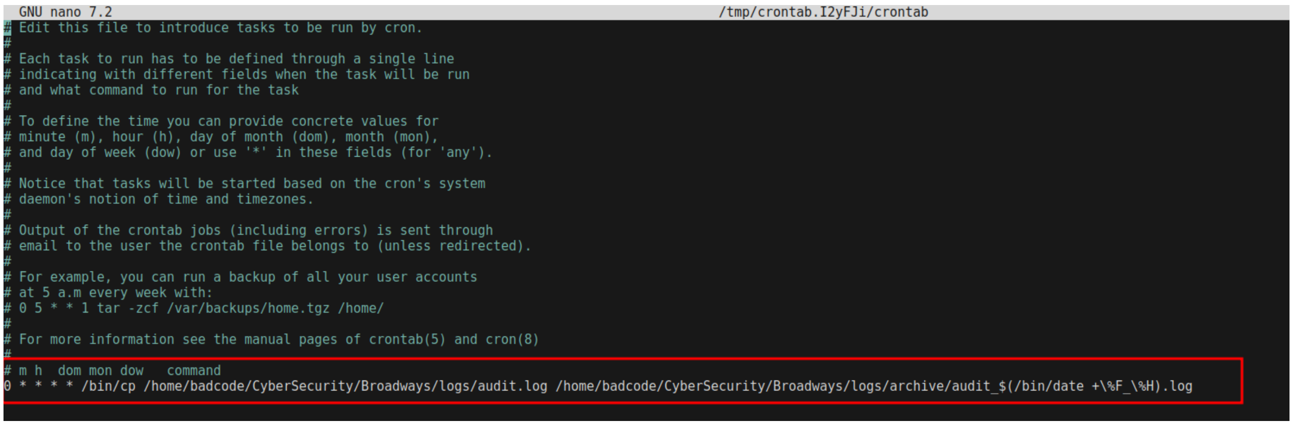


```
lok@Badcode:~$ id lok
uid=1012(lok) gid=1003(dev_team) groups=1003(dev_team),0(root),100(users),1005(audit_tea
m),1012(lok)
lok@Badcode:~$ sudo touch  /home/badcode/CyberSecurity/Broadways/LinuxChallenge/file.txt
[sudo] password for lok:
Sorry, try again.
[sudo] password for lok:
lok is not in the sudoers file.
lok@Badcode:~$  touch  /home/badcode/CyberSecurity/Broadways/LinuxChallenge/file.txt
lok@Badcode:~$ ls /home/badcode/CyberSecurity/Broadways/LinuxChallenge
audit.log  dev_notes.txt  file.txt
lok@Badcode:~$ ls -l /home/badcode/CyberSecurity/Broadways/LinuxChallenge
total 8
-rw-rw-r-- 1 mahesh audit_team 49 Jan  6 09:26 audit.log
-rw-rw-r-- 1 jatil  dev_team   25 Jan  4 16:57 dev_notes.txt
-rw-r--r-- 1 lok    dev_team    0 Jan 10 17:42 file.txt
lok@Badcode:~$
```

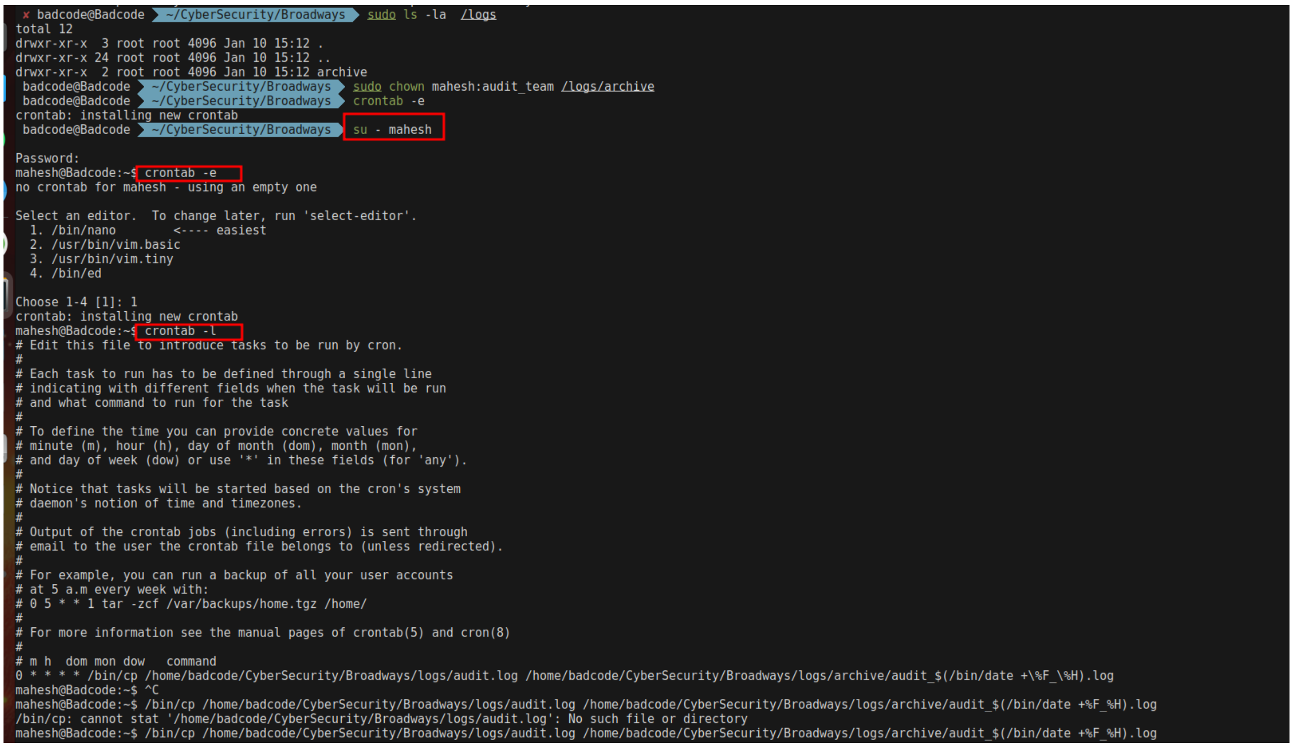
Part 6: Cron Jobs and Aliases

In this step, an automated backup mechanism was configured for the audit.log file. A cron job was set up for the user Mahesh to run hourly. Each execution of the cron job creates a backup of audit.log and saves it with a timestamped filename in the directory /home/badcode/CyberSecurity/Broadways/logs/archive/.

This ensures that a historical record of the audit logs is maintained automatically. Automating the backup process minimizes the risk of data loss, supports auditing and accountability requirements, and provides an easy way to track and restore logs if needed.



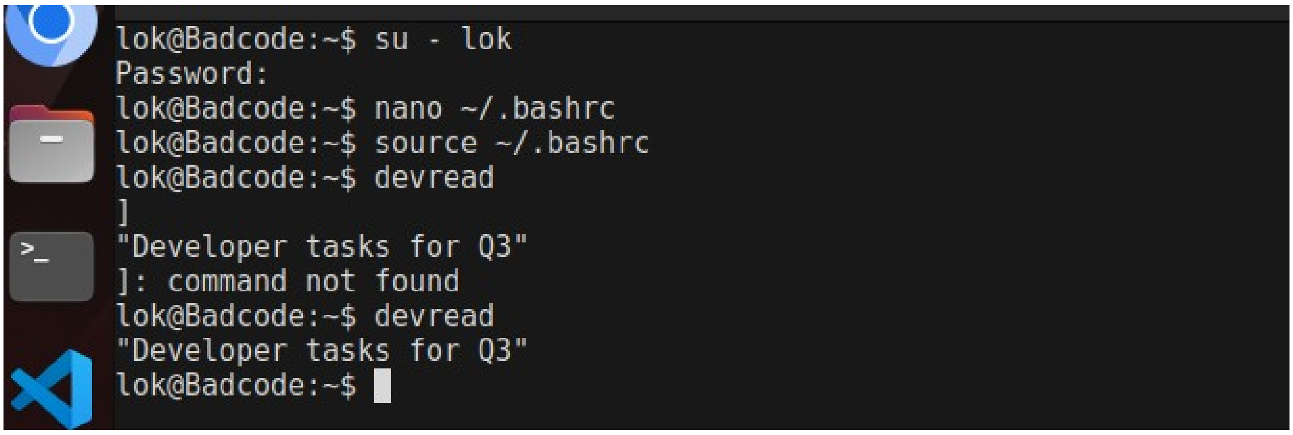
```
GNU nano 7.2                                                                                  /tmp/crontab.I2yFJi/crontab
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
0 * * * * /bin/cp /home/badcode/CyberSecurity/Broadways/logs/audit.log /home/badcode/CyberSecurity/Broadways/logs/archive/audit_$(/bin/date +\%F_\%H).log
```



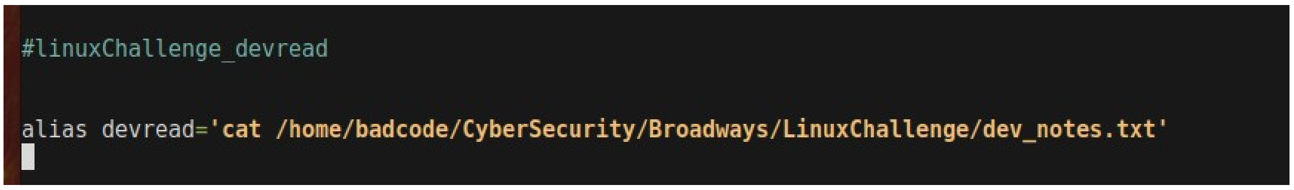
```
 x badcode@Badcode  ~/CyberSecurity/Broadways  sudo ls -la  /logs
total 12
drwxr-xr-x  3 root root 4096 Jan 10 15:12 .
drwxr-xr-x 24 root root 4096 Jan 10 15:12 ..
drwxr-xr-x  2 root root 4096 Jan 10 15:12 archive
 badcode@Badcode  ~/CyberSecurity/Broadways  sudo chown mahesh:audit_team /logs/archive
 badcode@Badcode  ~/CyberSecurity/Broadways  crontab -e
crontab: installing new crontab
 badcode@Badcode  ~/CyberSecurity/Broadways  su - mahesh

Password:
mahesh@Badcode:~$ crontab -e
no crontab for mahesh - using an empty one

Select an editor.  To change later, run 'select-editor'.
  1. /bin/nano        <---- easiest
  2. /usr/bin/vim.basic
  3. /usr/bin/vim.tiny
  4. /bin/ed

Choose 1-4 [1]: 1
crontab: installing new crontab
mahesh@Badcode:~$ crontab -l
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
0 * * * * /bin/cp /home/badcode/CyberSecurity/Broadways/logs/audit.log /home/badcode/CyberSecurity/Broadways/logs/archive/audit_$(/bin/date +\%F_\%H).log
mahesh@Badcode:~$ ^C
mahesh@Badcode:~$ /bin/cp /home/badcode/CyberSecurity/Broadways/logs/audit.log /home/badcode/CyberSecurity/Broadways/logs/archive/audit_$(/bin/date +%F_%H).log
/bin/cp: cannot stat '/home/badcode/CyberSecurity/Broadways/logs/audit.log': No such file or directory
mahesh@Badcode:~$ /bin/cp /home/badcode/CyberSecurity/Broadways/logs/audit.log /home/badcode/CyberSecurity/Broadways/logs/archive/audit_$(/bin/date +%F_%H).log
```

In this step, an alias named `devread` was created for the user `Lok` to simplify access to the development notes. The alias allows Lok to quickly view the dev notes without typing the full file path each time. To ensure that the alias remains available in all future sessions, it was added to Lok's `.bashrc` file.

By adding it to `.bashrc`, the alias is automatically loaded whenever Lok starts a new terminal session, improving efficiency and streamlining workflow for accessing important development information.



```
lok@Badcode:~$ su - lok
Password:
lok@Badcode:~$ nano ~/.bashrc
lok@Badcode:~$ source ~/.bashrc
lok@Badcode:~$ devread
]
"Developer tasks for Q3"
]: command not found
lok@Badcode:~$ devread
"Developer tasks for Q3"
lok@Badcode:~$
```

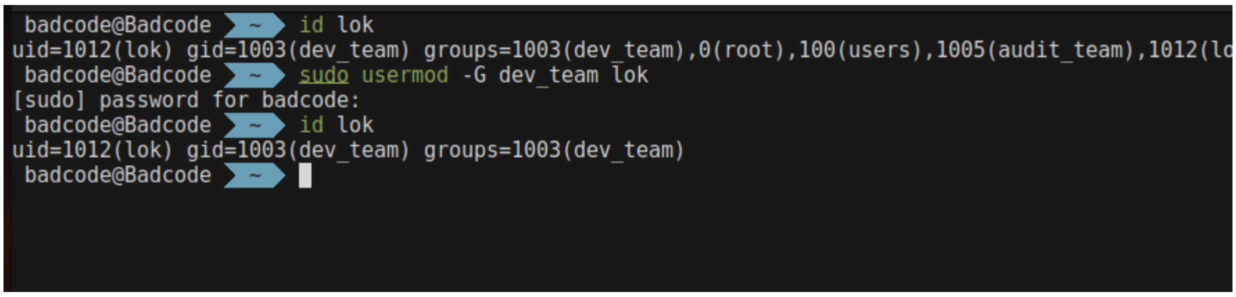


```
#linuxChallenge_devread

alias devread='cat /home/badcode/CyberSecurity/Broadways/LinuxChallenge/dev_notes.txt'
```

Part 7: Group Reassignment & Cleanup

In this step, the user `Lok` was reassigned to be a member of only the `dev_team`. Removing Lok from other groups affects the permissions he inherited from those groups. Specifically, any access that Lok had to files or directories through his previous group memberships is revoked.

For example, if Lok previously had access to certain directories or files via another group (such as `audit_team`), he would lose that access after being removed from the group. This reinforces controlled access and ensures that users can only interact with resources relevant to their current team, maintaining security and adherence to the principle of least privilege.

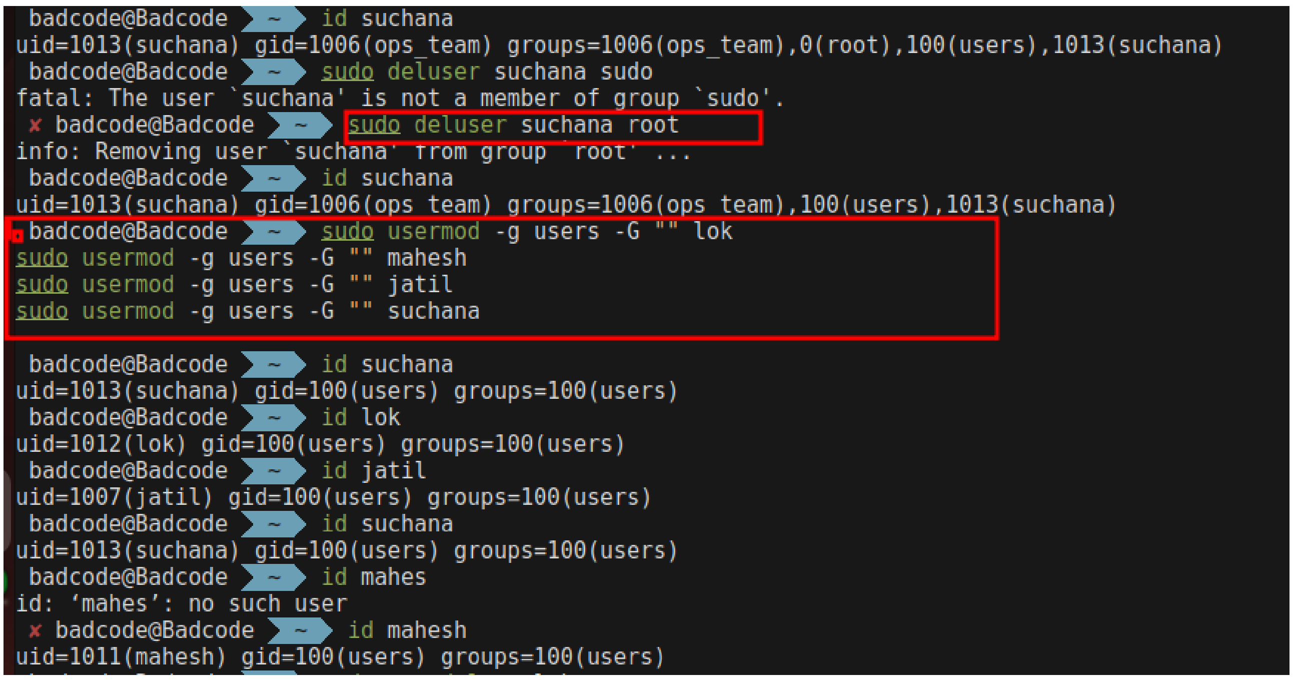


```
 badcode@Badcode  ~  id lok
uid=1012(lok) gid=1003(dev_team) groups=1003(dev_team),0(root),100(users),1005(audit_team),1012(lo
 badcode@Badcode  ~  sudo usermod -G dev_team lok
[sudo] password for badcode:
 badcode@Badcode  ~  id lok
uid=1012(lok) gid=1003(dev_team) groups=1003(dev_team)
 badcode@Badcode  ~ 
```

Removing Suchana from the sudo group:
The user `Suchana` was removed from the `sudo` group to revoke her administrative privileges.

This ensures that she can no longer execute commands with elevated privileges, enforcing stricter access control and reducing the risk of unauthorized system changes.

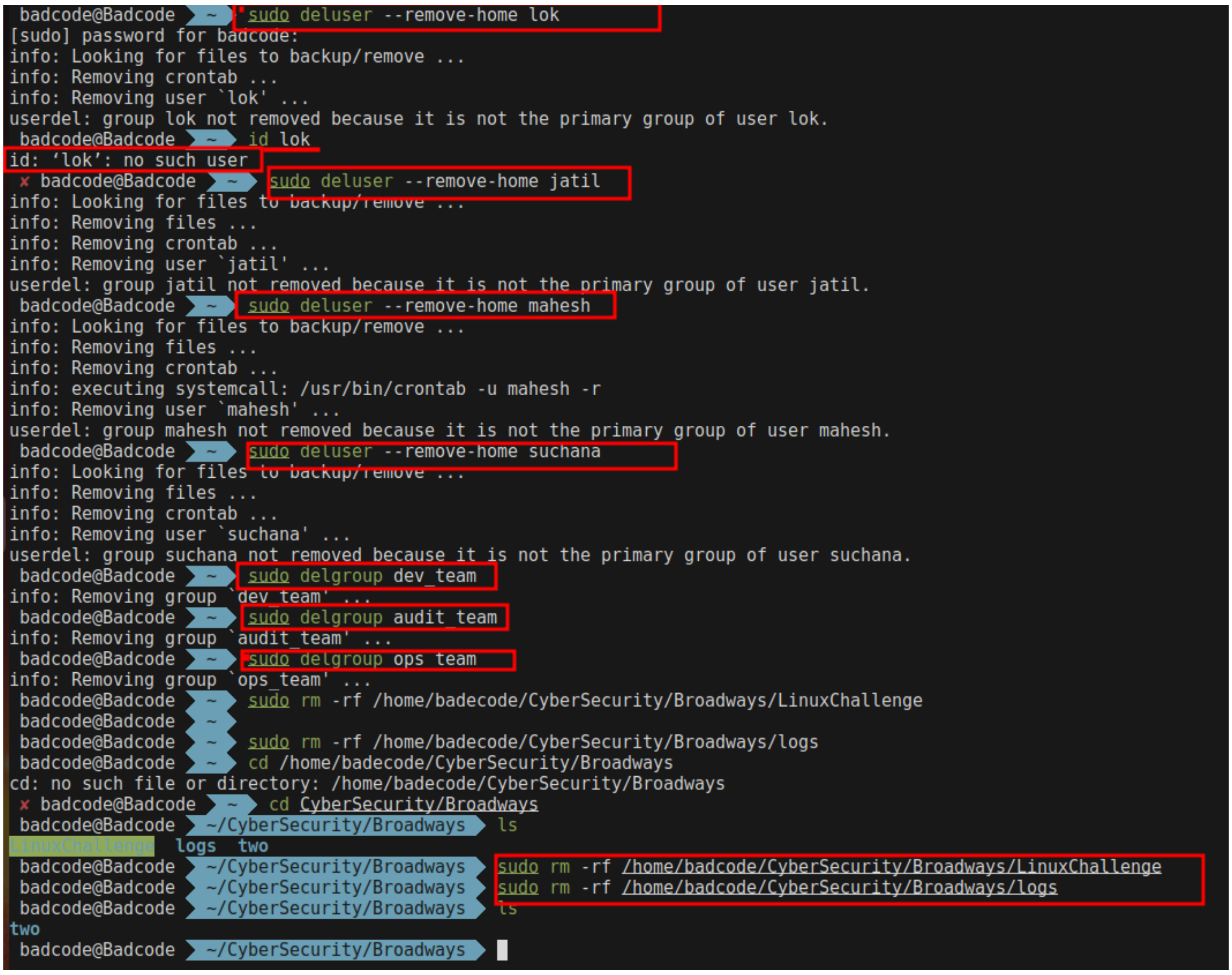
Resetting all users' group memberships:
All users were removed from their secondary groups, and their primary group was set to `users`. This step simplifies group management and ensures a consistent baseline for permissions. Any access previously granted through secondary groups is revoked, limiting users to the minimum required permissions associated with the `users` group. This cleanup helps maintain a secure and organized environment, preventing unintended access or privilege escalation.



```
 badcode@Badcode  ~  id suchana
uid=1013(suchana) gid=1006(ops_team) groups=1006(ops_team),0(root),100(users),1013(suchana)
 badcode@Badcode  ~  sudo deluser suchana sudo
fatal: The user `suchana' is not a member of group `sudo'.
 ✘ badcode@Badcode  ~  sudo deluser suchana root
info: Removing user `suchana' from group `root' ...
 badcode@Badcode  ~  id suchana
uid=1013(suchana) gid=1006(ops_team) groups=1006(ops_team),100(users),1013(suchana)
 badcode@Badcode  ~  sudo usermod -g users -G "" lok
sudo usermod -g users -G "" mahesh
sudo usermod -g users -G "" jatil
sudo usermod -g users -G "" suchana

 badcode@Badcode  ~  id suchana
uid=1013(suchana) gid=100(users) groups=100(users)
 badcode@Badcode  ~  id lok
uid=1012(lok) gid=100(users) groups=100(users)
 badcode@Badcode  ~  id jatil
uid=1007(jatil) gid=100(users) groups=100(users)
 badcode@Badcode  ~  id suchana
uid=1013(suchana) gid=100(users) groups=100(users)
 badcode@Badcode  ~  id mahes
id: 'mahes': no such user
 ✘ badcode@Badcode  ~  id mahesh
uid=1011(mahesh) gid=100(users) groups=100(users)
```

Deleting all users:
All users created during the exercise were deleted along with their home directories. This ensures that no unauthorized accounts remain on the system and that all personal files and configurations associated with these users are removed, maintaining system security and cleanliness.

Deleting all groups and directories:
All groups created for the exercise, as well as directories such as project or log directories, were deleted. This final cleanup ensures that no residual permissions, files, or group associations remain, leaving the system in a clean and secure state.



```
 badcode@Badcode  ~  sudo deluser --remove-home lok
[sudo] password for badcode:
info: Looking for files to backup/remove ...
info: Removing crontab ...
info: Removing user `lok' ...
userdel: group lok not removed because it is not the primary group of user lok.
 badcode@Badcode  ~  id lok
id: 'lok': no such user
 ✘ badcode@Badcode  ~  sudo deluser --remove-home jatil
info: Looking for files to backup/remove ...
info: Removing files ...
info: Removing crontab ...
info: Removing user `jatil' ...
userdel: group jatil not removed because it is not the primary group of user jatil.
 badcode@Badcode  ~  sudo deluser --remove-home mahesh
info: Looking for files to backup/remove ...
info: Removing files ...
info: Removing crontab ...
info: executing systemcall: /usr/bin/crontab -u mahesh -r
info: Removing user `mahesh' ...
userdel: group mahesh not removed because it is not the primary group of user mahesh.
 badcode@Badcode  ~  sudo deluser --remove-home suchana
info: Looking for files to backup/remove ...
info: Removing files ...
info: Removing crontab ...
info: Removing user `suchana' ...
userdel: group suchana not removed because it is not the primary group of user suchana.
 badcode@Badcode  ~  sudo delgroup dev_team
info: Removing group `dev_team' ...
 badcode@Badcode  ~  sudo delgroup audit_team
info: Removing group `audit_team' ...
 badcode@Badcode  ~  sudo delgroup ops_team
info: Removing group `ops_team' ...
 badcode@Badcode  ~  sudo rm -rf /home/badecode/CyberSecurity/Broadways/LinuxChallenge
 badcode@Badcode  ~ 
 badcode@Badcode  ~  sudo rm -rf /home/badecode/CyberSecurity/Broadways/logs
 badcode@Badcode  ~  cd /home/badecode/CyberSecurity/Broadways
cd: no such file or directory: /home/badecode/CyberSecurity/Broadways
 ✘ badcode@Badcode  ~  cd CyberSecurity/Broadways
 badcode@Badcode  ~/CyberSecurity/Broadways  ls
LinuxChallenge  logs  two
 badcode@Badcode  ~/CyberSecurity/Broadways  sudo rm -rf /home/badcode/CyberSecurity/Broadways/LinuxChallenge
 badcode@Badcode  ~/CyberSecurity/Broadways  sudo rm -rf /home/badcode/CyberSecurity/Broadways/logs
 badcode@Badcode  ~/CyberSecurity/Broadways  ls
two
 badcode@Badcode  ~/CyberSecurity/Broadways 
```

Project Conclusion

Through this project, I gained practical experience in managing Linux users, groups, and file permissions, while implementing secure system administration practices. I successfully created and organized users and groups, applied precise ownership and permission settings to files and directories, and enforced role-based access control using SetGID and sudo rules.

The project also allowed me to test and validate security measures by simulating real-world scenarios, such as restricting unauthorized access, controlling administrative privileges, and automating tasks like log backups with cron jobs. Creating persistent aliases further improved workflow efficiency and demonstrated the importance of user-centric configurations.

Finally, performing a full cleanup by reassigning groups, removing sudo privileges, and deleting temporary users and directories reinforced best practices for maintaining system hygiene and minimizing security risks.

Overall, this project enhanced my understanding of Linux administration, strengthened my ability to implement secure and organized systems, and provided hands-on experience in managing users, permissions, and automated administrative tasks in a professional environment.