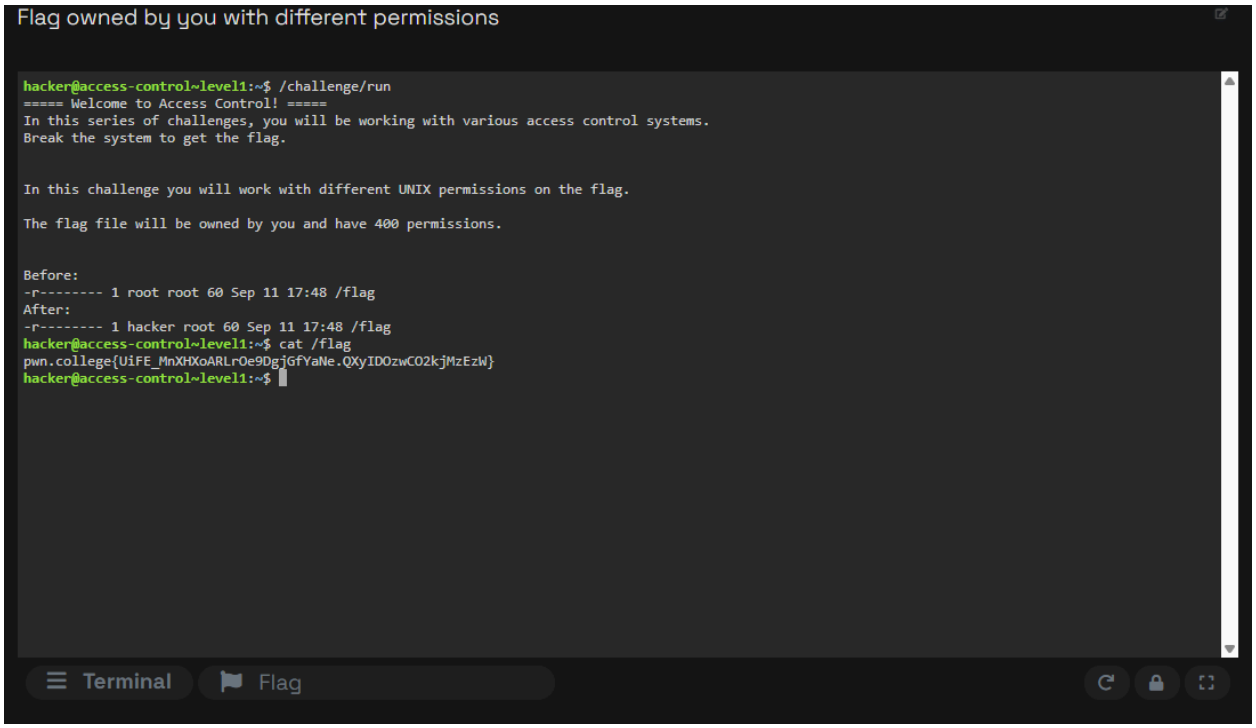
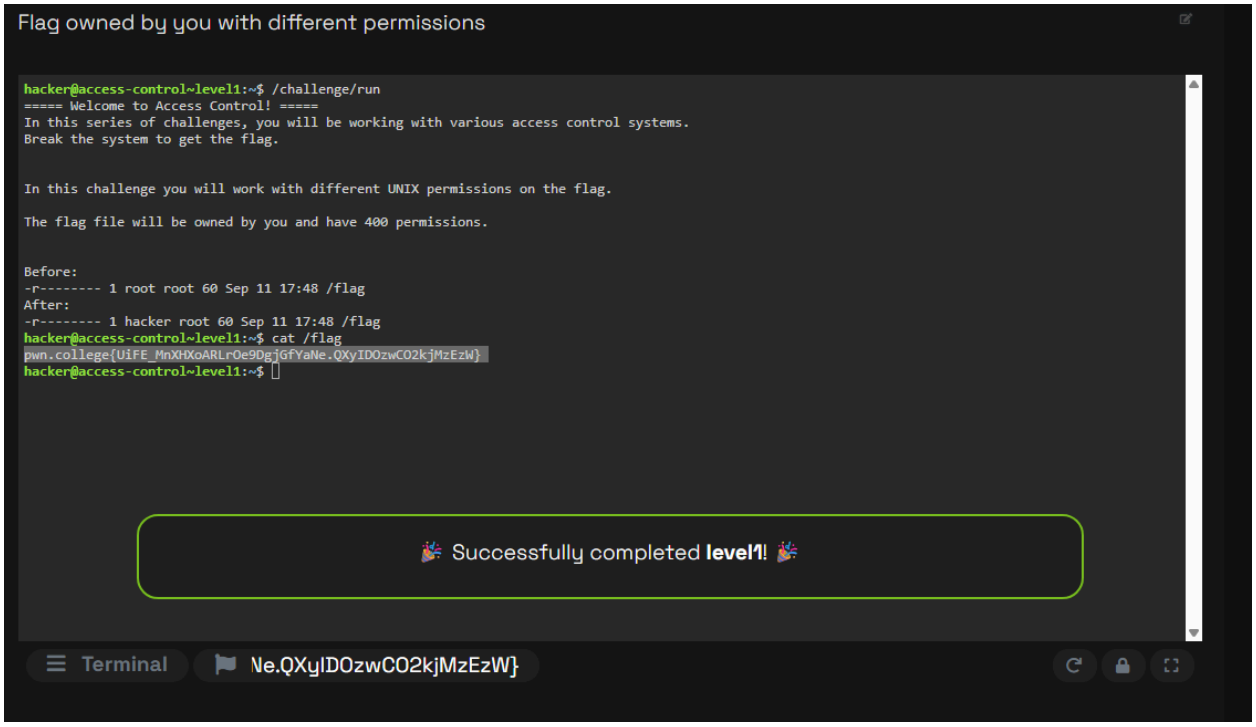
Jaleel Williamson
jayw-713
CSCI 400 Lab 5
9/18/25

Intro to Cybersecurity: Access Control
Unix Permissions
https://pwn.college/intro-to-cybersecurity/access-control/

- **Level 1**

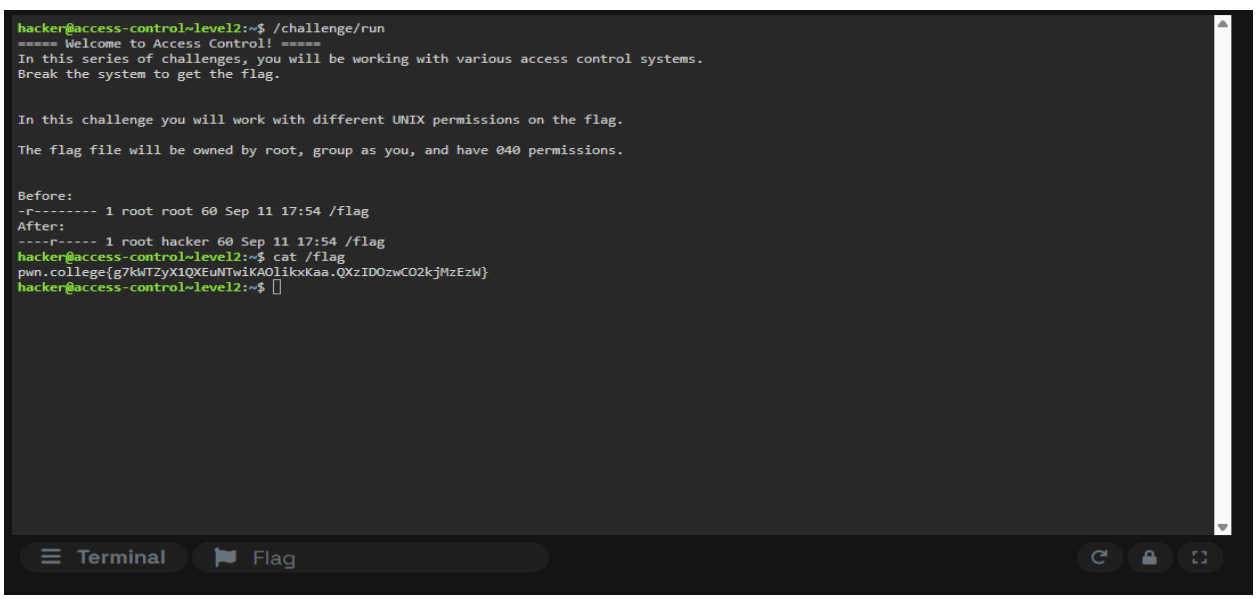


Screenshot taken before successful capture.



After running the challenge binary, I found that the **/flag file** was now owned by me with read-only permissions, so I could directly access it using **cat /flag**. It felt straightforward but reinforced my understanding of how UNIX file permissions work, especially how ownership grants immediate access without needing complex bypasses. Luckily in this challenge we had **400 permissions**.

- **Level 2**



Screenshot taken before successful capture.

```
hacker@access-control~level2:~$ /challenge/run
===== Welcome to Access Control! =====
In this series of challenges, you will be working with various access control systems.
Break the system to get the flag.


In this challenge you will work with different UNIX permissions on the flag.

The flag file will be owned by root, group as you, and have 040 permissions.


Before:
-r-------- 1 root root 60 Sep 11 17:54 /flag
After:
----r----- 1 root hacker 60 Sep 11 17:54 /flag
hacker@access-control~level2:~$ cat /flag
pwn.college{g7kWTZyX1QXEuNTwiKAOlikxKaa.QXzIDOzwCO2kjMzEzW}
hacker@access-control~level2:~$
```

🎉 Successfully completed **level2**! 🎉

Terminal aa.QXzIDOzwCO2kjMzEzW}

After running the challenge binary, the **/flag file** had its **permissions** set to **040**, meaning only the group could read it, and since I was in the hacker group, I could directly access it using **cat /flag**. This highlighted how **group permissions** in UNIX can grant access even if the user isn't the owner.

- **Level 3**

```
hacker@access-control~level3:~$ /challenge/run
===== Welcome to Access Control! =====
In this series of challenges, you will be working with various access control systems.
Break the system to get the flag.


In this challenge you will work with different UNIX permissions on the flag.

The flag file will be owned by you and have 000 permissions.


Before:
-r-------- 1 root root 60 Sep 11 17:58 /flag
After:
---------- 1 hacker root 60 Sep 11 17:58 /flag
hacker@access-control~level3:~$ chmod 400 /flag
hacker@access-control~level3:~$ cat /flag
pwn.college{YBlqUX_utuvAfHpdoDnlv171edA.QX0IDOzwCO2kjMzEzW}
hacker@access-control~level3:~$
```

Terminal Flag

Screenshot taken before successful capture.

```
hacker@access-control~level3:~$ /challenge/run
===== Welcome to Access Control! =====
In this series of challenges, you will be working with various access control systems.
Break the system to get the flag.


In this challenge you will work with different UNIX permissions on the flag.

The flag file will be owned by you and have 000 permissions.


Before:
-r-------- 1 root root 60 Sep 11 17:58 /flag
After:
---------- 1 hacker root 60 Sep 11 17:58 /flag
hacker@access-control~level3:~$ chmod 400 /flag
hacker@access-control~level3:~$ cat /flag
pwn.college{YBlqUX_utuvAfHpdoOnlv171edA.QX0IDOzwCO2kjMzEzW}
hacker@access-control~level3:~$
```

🎉 Successfully completed **level3**! 🎉

Terminal ⚑ ɟA.QX0IDOzwCO2kjMzEzW}

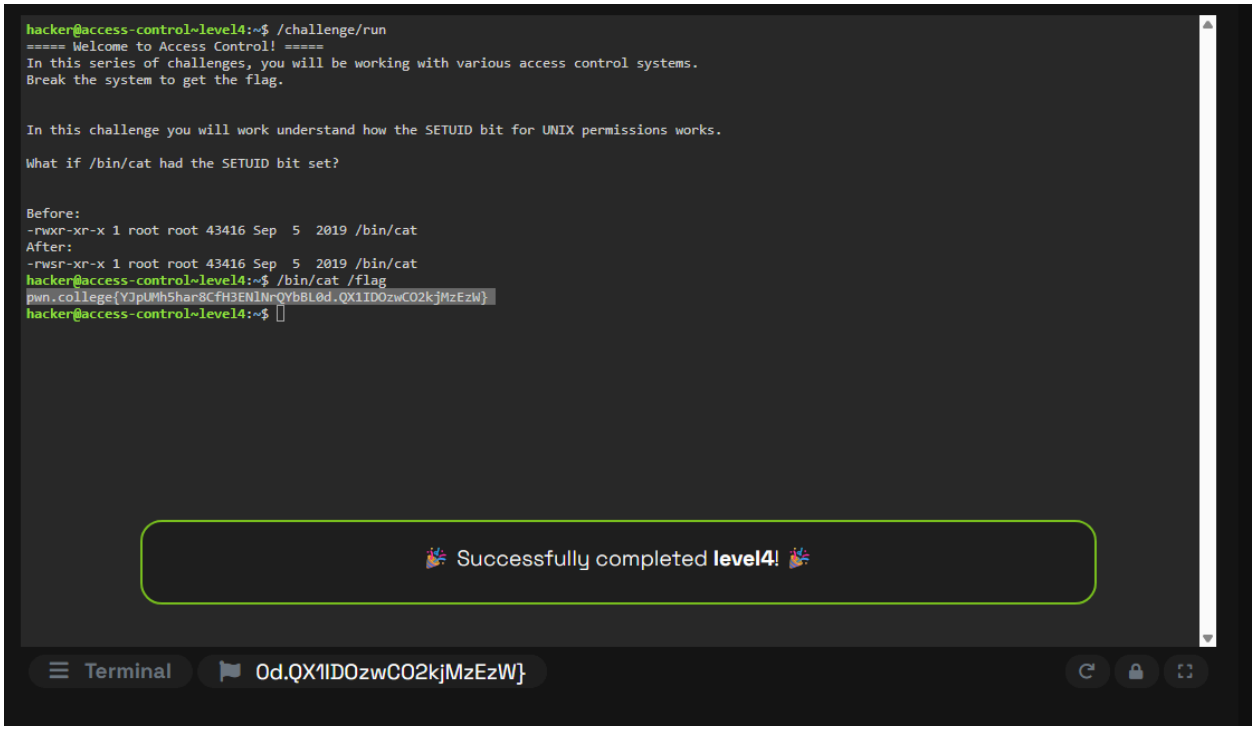
After running the challenge binary, the **/flag file** was owned by me but had **no permissions (000)**. Since I owned the file, I used **chmod** to add read permissions for myself with **chmod 400 /flag**. This allowed me to then read the flag directly using **cat /flag**. This demonstrated that ownership grants the ability to **modify** permissions, even when initial access is restricted.

- **Level 4**

```
hacker@access-control~level4:~$ /challenge/run
===== Welcome to Access Control! =====
In this series of challenges, you will be working with various access control systems.
Break the system to get the flag.


In this challenge you will work understand how the SETUID bit for UNIX permissions works.

What if /bin/cat had the SETUID bit set?


Before:
-rwxr-xr-x 1 root root 43416 Sep  5  2019 /bin/cat
After:
-rwsr-xr-x 1 root root 43416 Sep  5  2019 /bin/cat
hacker@access-control~level4:~$ /bin/cat /flag
pwn.college{YJpUMh5har8CfH3ENlNrQYbBL0d.QX1IDOzwCO2kjMzEzW}
hacker@access-control~level4:~$
```

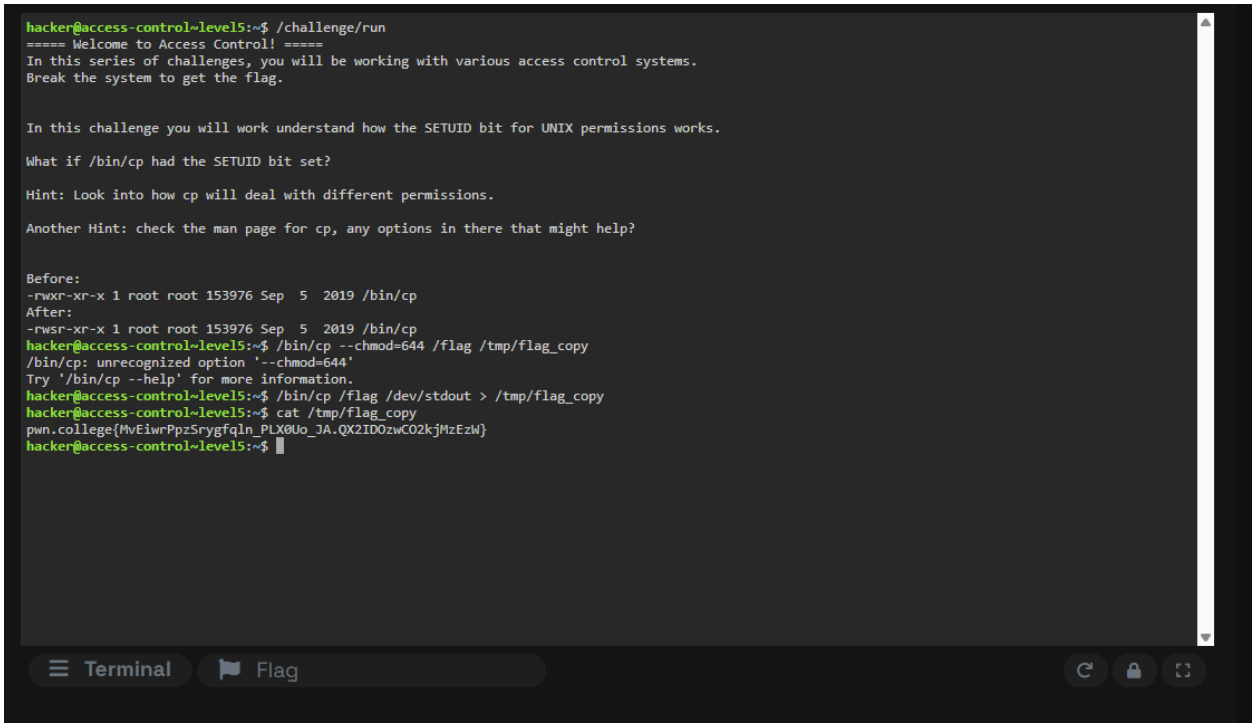
Terminal ⚑ Flag

Screenshot taken before successful capture.

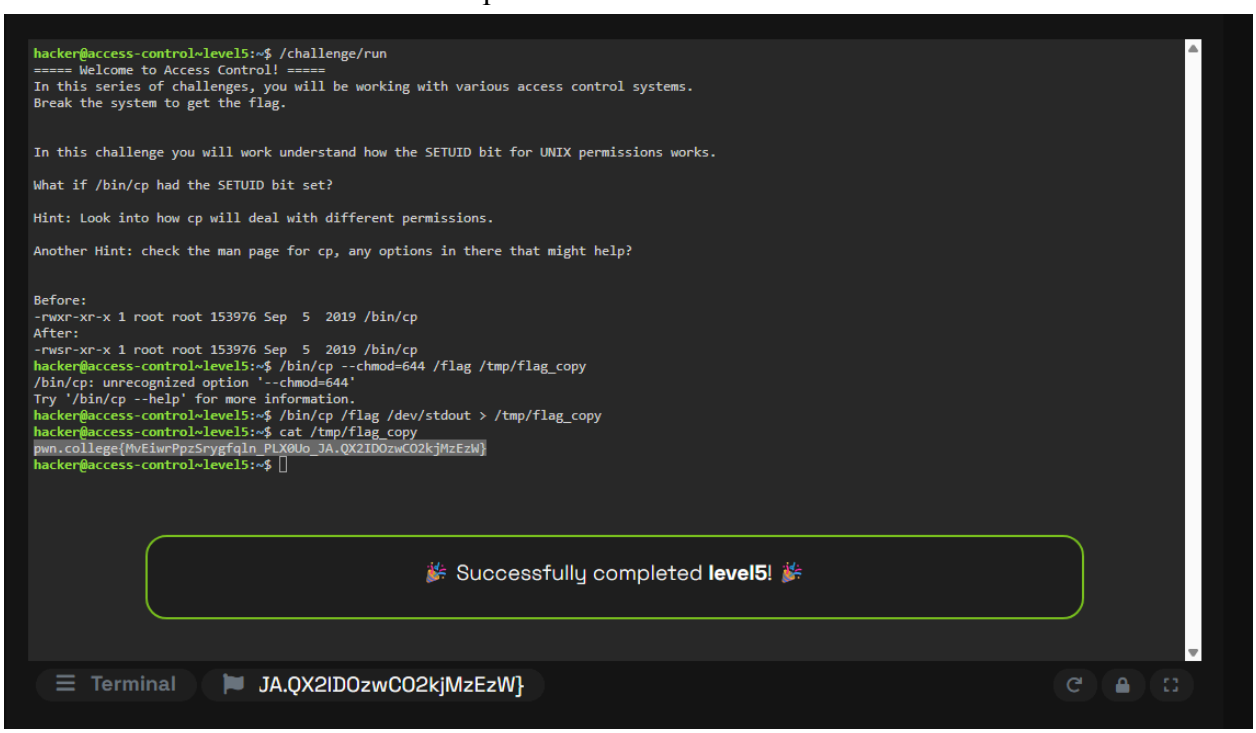


After running the challenge binary, **/bin/cat** had the **SETUID** bit set, meaning it would execute with root privileges. Since the flag file (likely /flag) is owned by root and has restricted permissions, I used **/bin/cat /flag** to read it. This worked because the **SETUID-enabled** cat ran as root, bypassing the permission checks and allowing me to access the flag directly. This demonstrated the power of **SETUID** in granting elevated privileges through specific binaries.

- **Level 5**



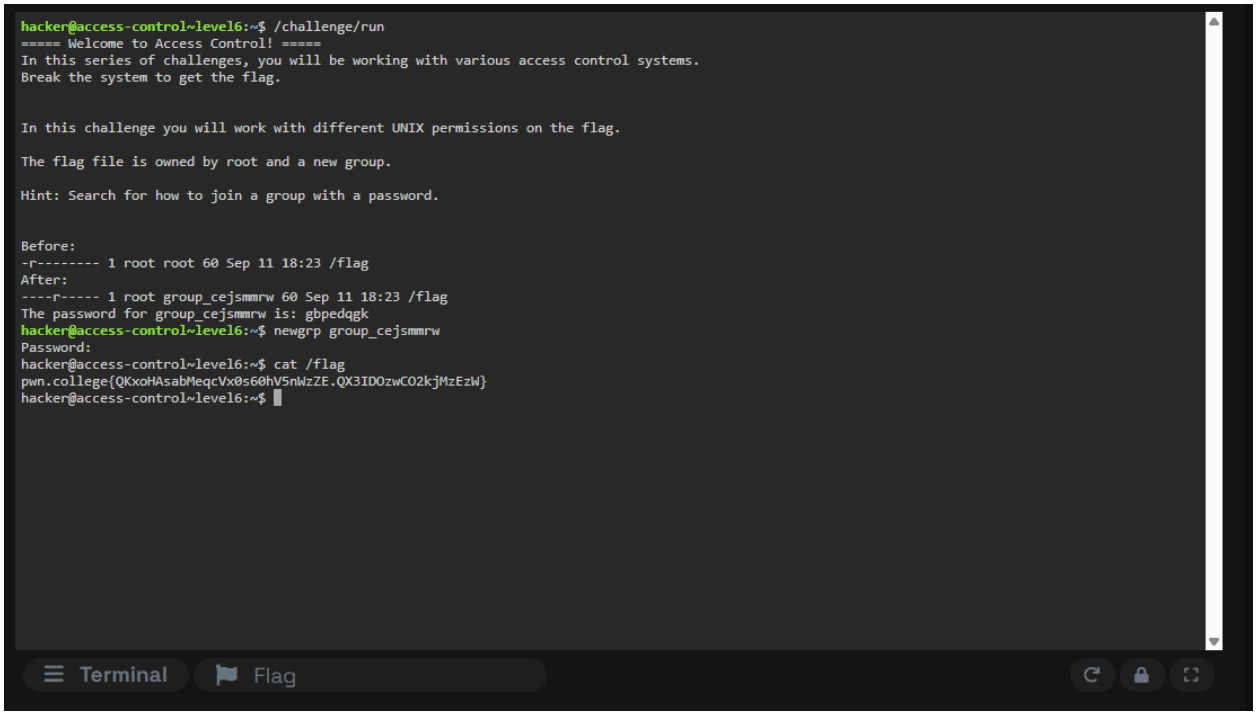
Screenshot taken before successful capture.



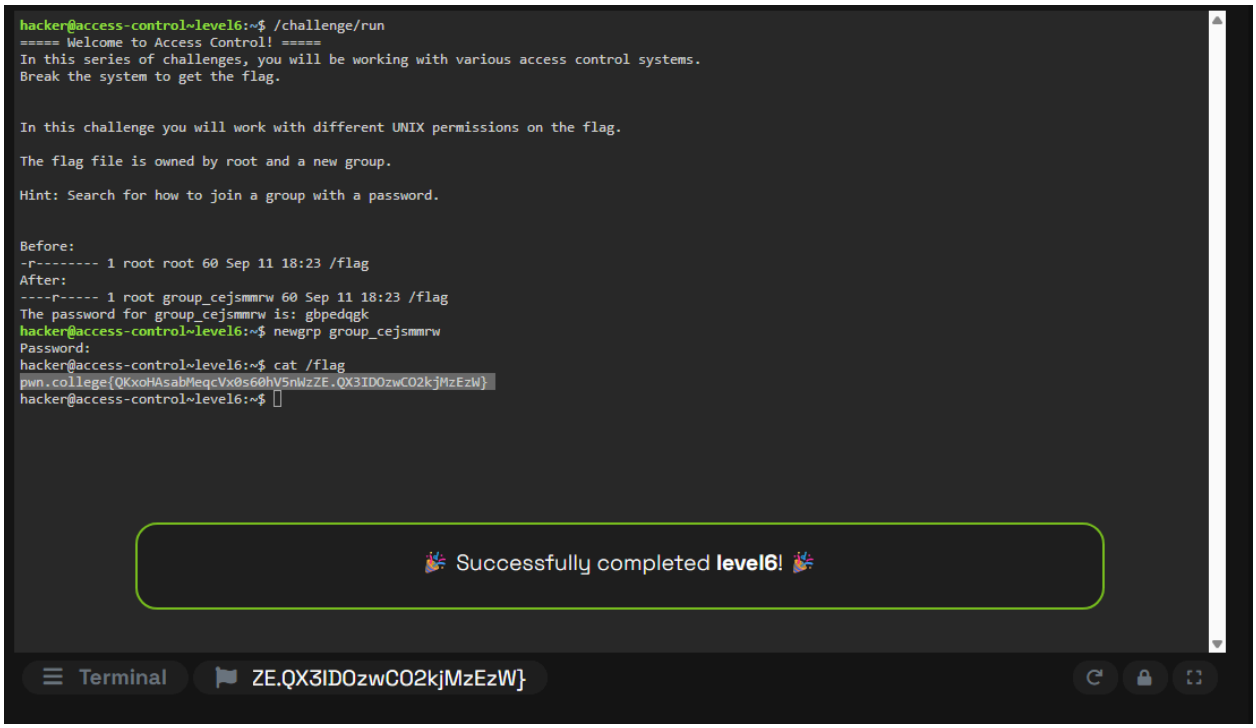
After running the challenge binary, **/bin/cp** had the **SETUID** bit set, allowing it to run with root privileges. To access the flag, I used **/bin/cp** to copy the contents of /flag to standard output and then redirected that output to a file in **/tmp**, which I could read. This worked because the **SETUID-enabled** cp could read the restricted **/flag** file as root, and the redirection created a file

owned by me with default permissions. This approach bypassed the permission issues since the copied file was created under my user account, making it accessible without needing to modify permissions directly.

- **Level 6**

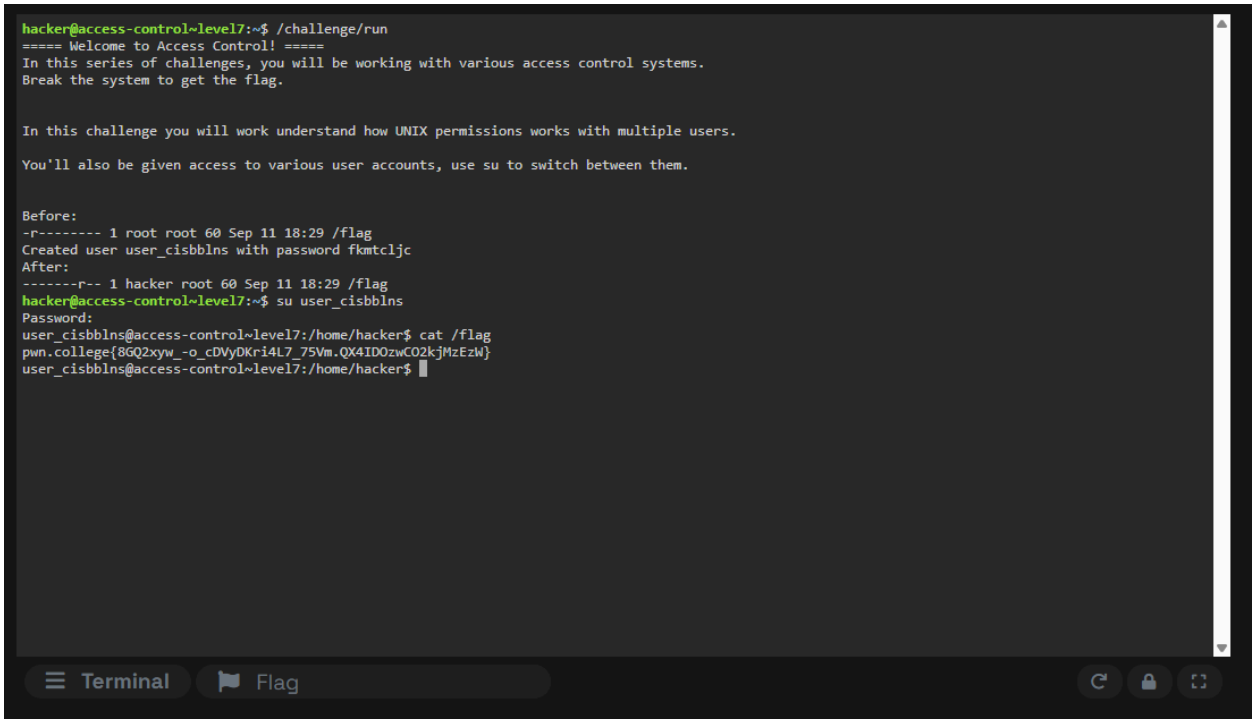


Screenshot taken before successful capture.

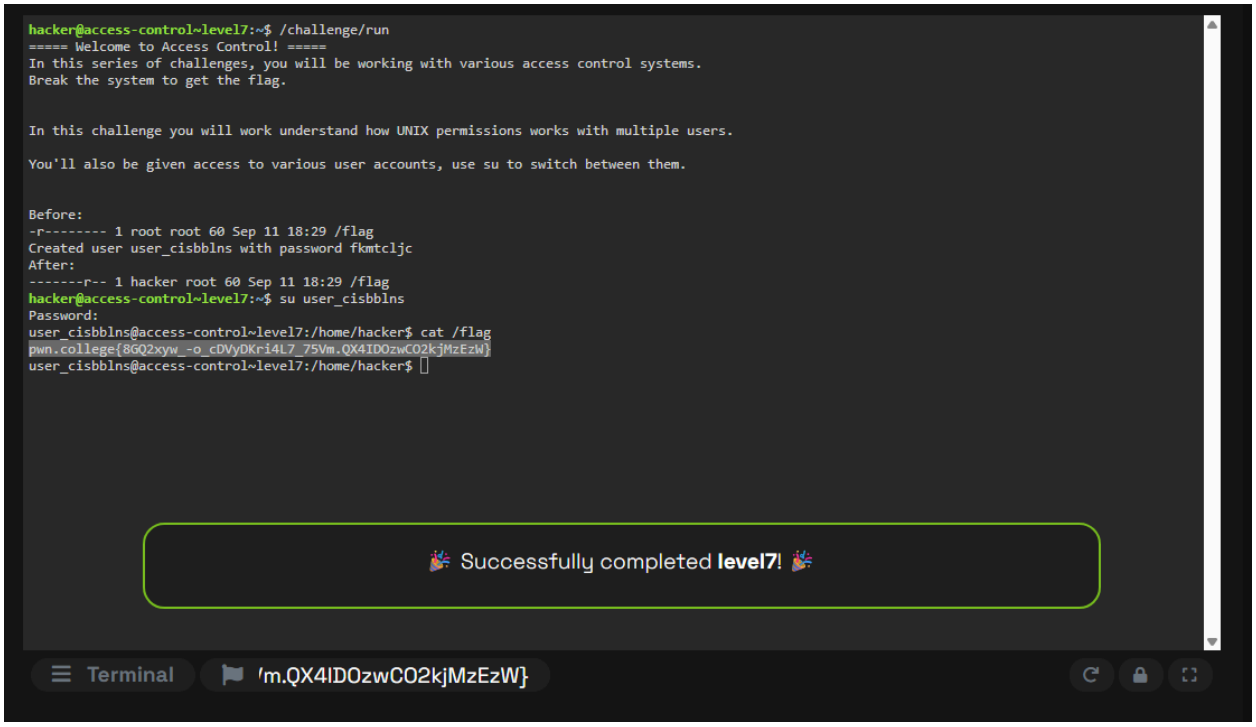


After running the challenge binary, the **/flag** file had its permissions set to allow only the group **group_cejsmmrw** to read it. Since I was not initially a member of that group, I used the **newgrp** command with the provided password to switch to that group. After entering the password, I gained group membership and was able to read the flag directly using **cat /flag**. This demonstrated how group passwords can be used to gain access to restricted files without root intervention.

- **Level 7**

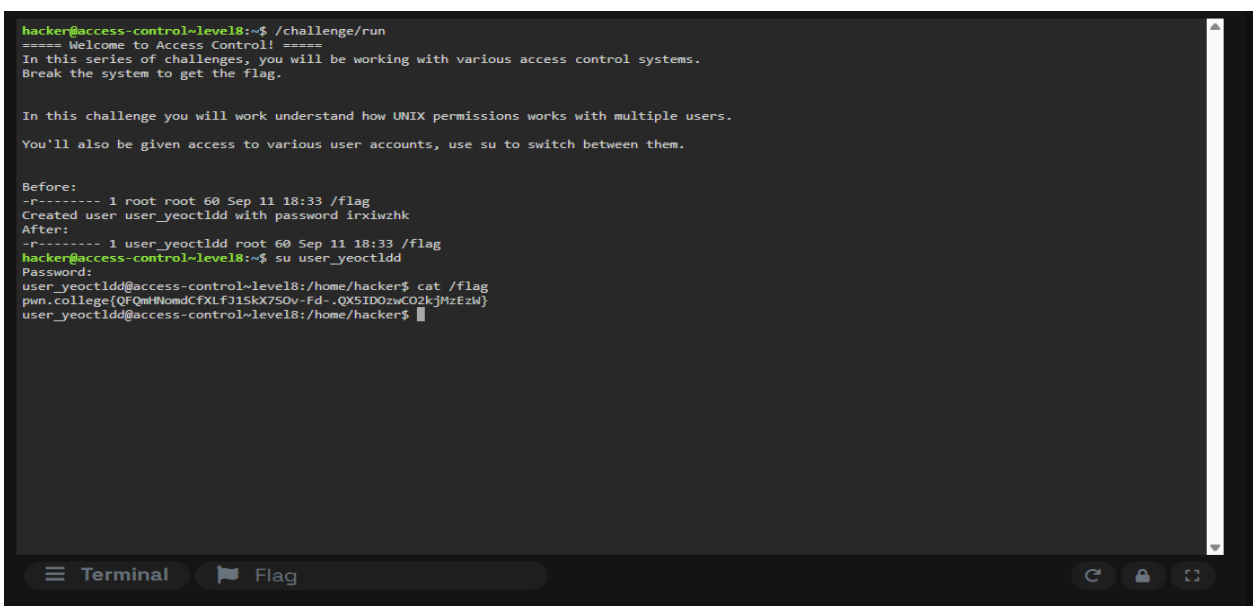


Screenshot taken before successful capture.

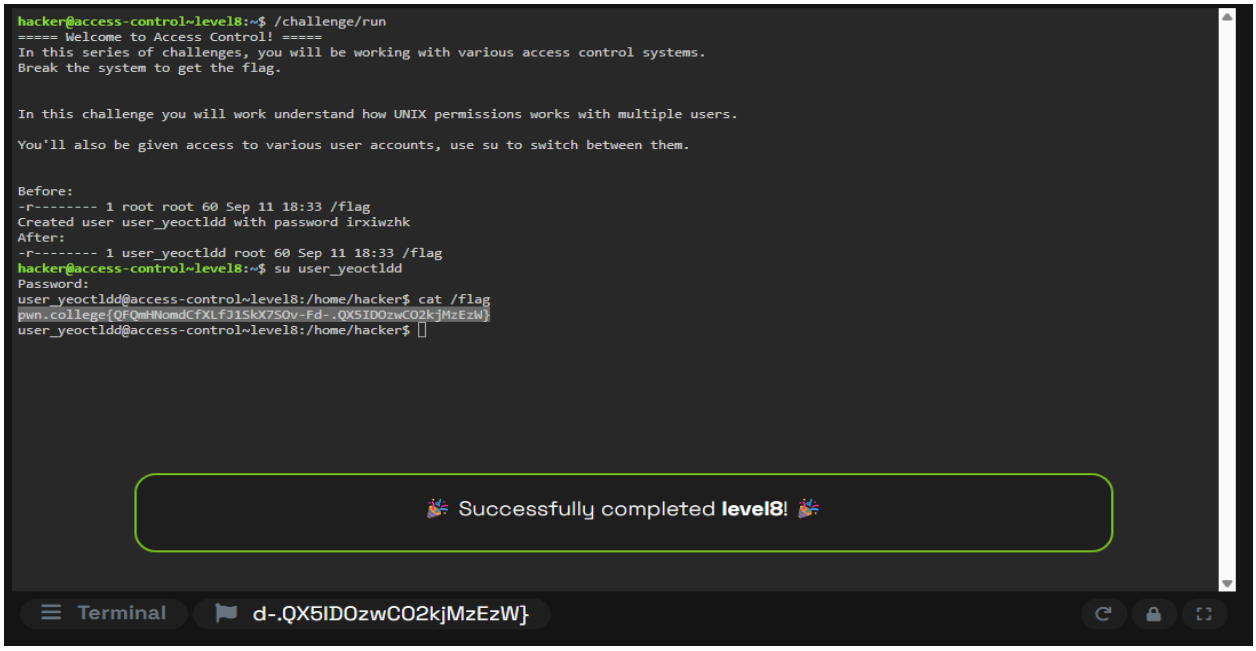


After running the challenge binary, the **/flag file** had permissions set to **-------r--**, meaning only others could read it. Since I was the **owner (hacker)** but without read permissions, I needed to access it as another user. I used the provided credentials to switch to **user_cisbblns** with the password **fkmtcljc** using **su user_cisbblns**. Once switched, I was able to read the flag directly with **cat /flag** because the file granted read access to others. This demonstrated how UNIX permissions can allow access through other users when appropriate permissions are set.

- **Level 8**

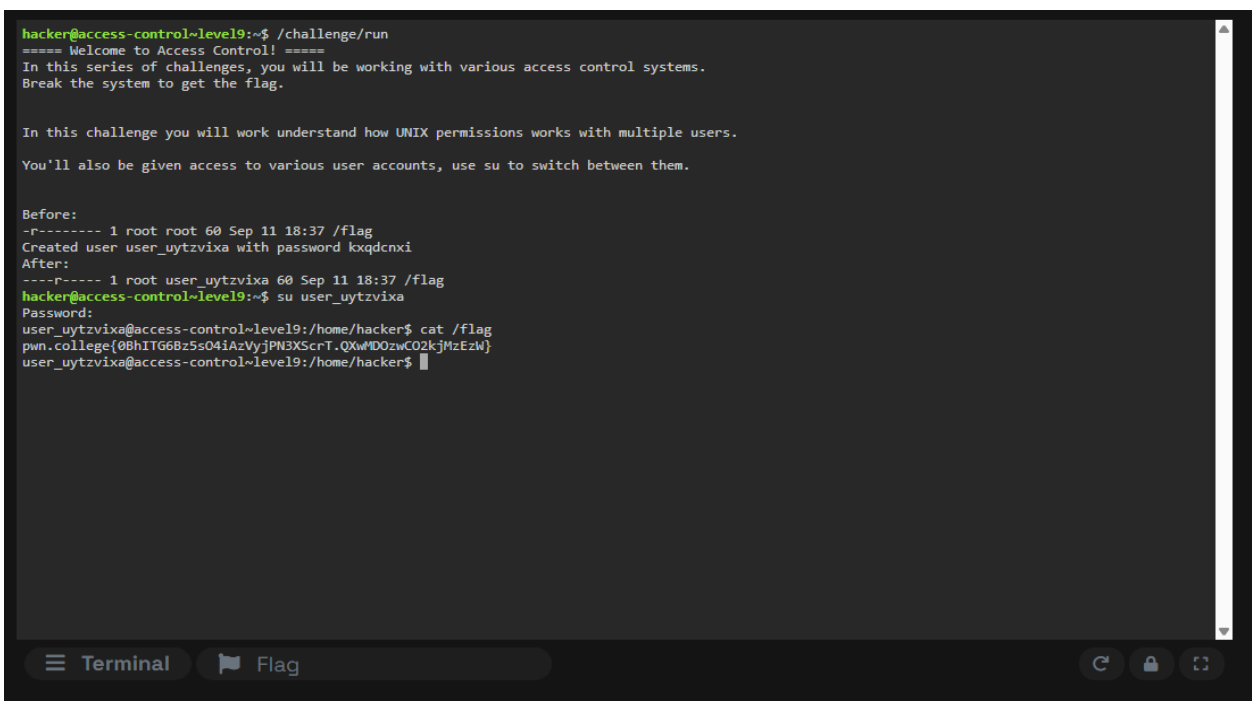


Screenshot taken before successful capture.

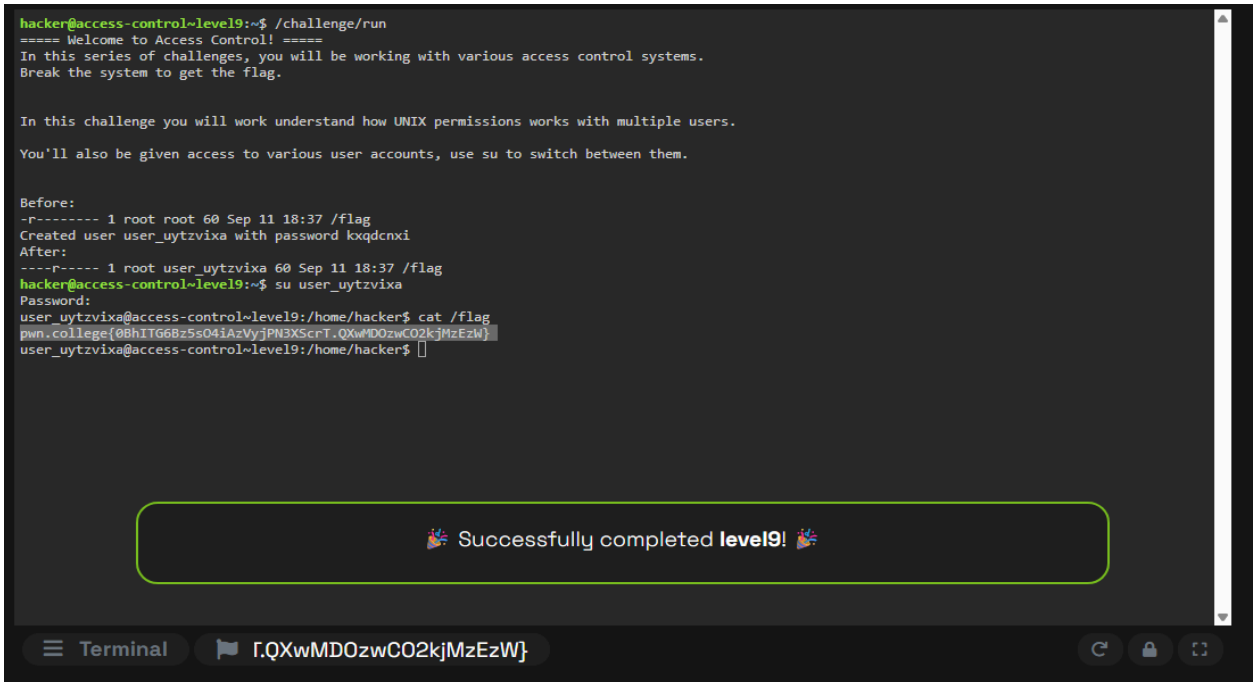


After running the challenge binary, the /flag file was owned by **user_yeoctldd** with read-only **permissions (400)**. Since I was logged in as hacker, I used **su user_yeoctldd** with the provided password **irxiwzhk** to switch to that user. Once switched, I was able to read the flag directly with **cat /flag** because I became the **owner** of the file. This demonstrated how switching users can grant access to files based on ownership and permissions.

- **Level 9**

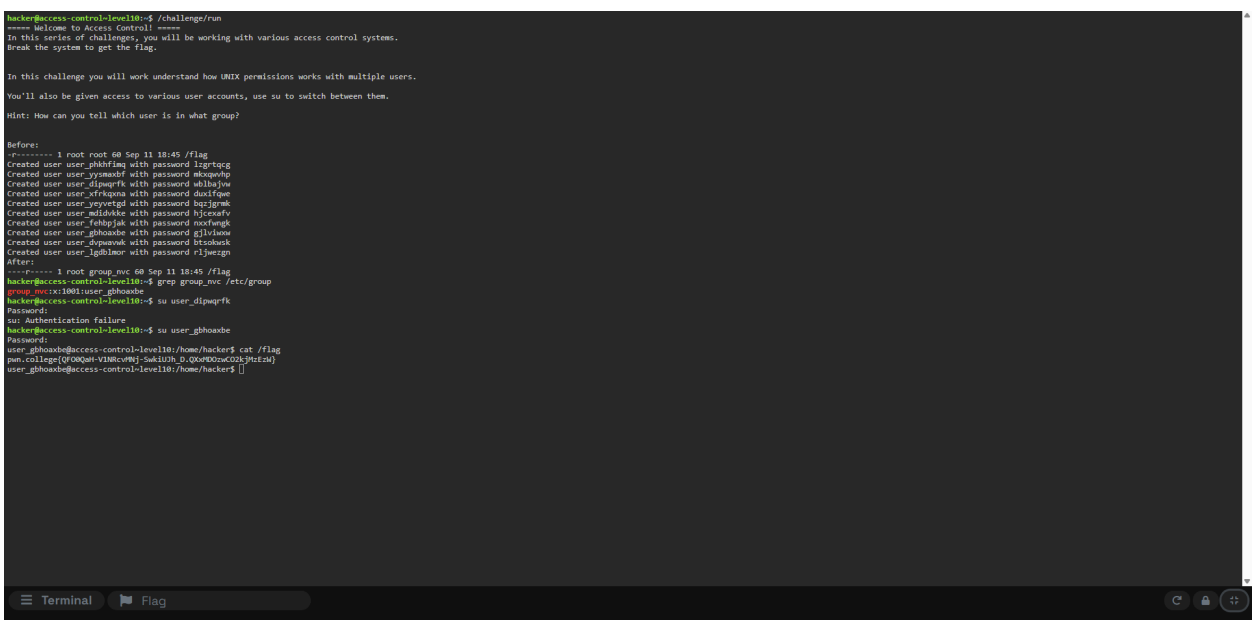


Screenshot taken before successful capture.



After running the challenge binary, the **/flag** file had its permissions set to allow only the group **user_uytzvixa** to read it. Since I was logged in as hacker, I used **su user_uytzvixa** with the provided password **kxqdcnxi** to switch to that user. Once switched, I was able to read the flag directly with **cat /flag** because **user_uytzvixa** is a member of the group **user_uytzvixa**, which had the necessary read permissions. This demonstrated how group-based permissions can be accessed by switching to a user account that belongs to the required group.

- **Level 10**



Screenshot taken before successful capture.

```
hacker@access-control~level10:~$ /challenge/run
===== Welcome to Access Control! =====
In this series of challenges, you will be working with various access control systems.
Break the system to get the flag.


In this challenge you will work understand how UNIX permissions works with multiple users.

You'll also be given access to various user accounts, use su to switch between them.

Hint: How can you tell which user is in what group?


Before:
-r-------- 1 root root 60 Sep 11 18:45 /flag
Created user user_phkhfimq with password lzgrtqcg
Created user user_yysmaxbf with password mkoqwvhp
Created user user_dipwqrfk with password wblbajvw
Created user user_xfrkqxna with password duxifqwe
Created user user_yeyvetgd with password bqzjgrmk
Created user user_mdldvkke with password hjcsxafv
Created user user_fehbpjak with password nxxfwngk
Created user user_gbhoaxbe with password gjlviwxw
Created user user_dvgwaavwk with password btsoaxsk
Created user user_lgdblmor with password rljwezgn
After:
----r----- 1 root group_nvc 60 Sep 11 18:45 /flag
hacker@access-control~level10:~$ grep group_nvc /etc/group
group_nvc:x:1001:user_gbhoaxbe
hacker@access-control~level10:~$ su user_dipwqrfk
Password:
su: Authentication failure
hacker@access-control~level10:~$ su user_gbhoaxbe
Password:
user_gbhoaxbe@access-control~level10:/home/hacker$ cat /flag
pwn.college{QfO0QwH-V1NBcvMMj-5wkiU3h_D.QXxMDOzwCO2kjMzEzW}
user_gbhoaxbe@access-control~level10:/home/hacker$
```

🎉 Successfully completed **level10**! 🎉

Terminal | D.QXxMDOzwCO2kjMzEzW}

After running the challenge binary, the **/flag file** was set to be readable only by the group **group_nvc**. To determine which user belongs to this group, I used **grep group_nvc /etc/group**, which revealed that **user_gbhoaxbe** is a member of **group_nvc**. I then switched to this user using the provided password **(gjlviwxw)** and read the flag with **cat /flag**. This demonstrated how to identify group membership and switch users to access files based on group permissions.

- **Level 11**

```
hacker@access-control~level11:~$ /challenge/run
===== Welcome to Access Control! =====
In this series of challenges, you will be working with various access control systems.
Break the system to get the flag.


In this challenge you will work understand how UNIX permissions for directories work with multiple users.

You'll be given access to various user accounts, use su to switch between them.


Created user user_rakstfeu with password wocsjuov
Created user user_knxbwkvf with password izumtiva
A copy of the flag has been placed somewhere in /tmp:
total 36
drwxrwxrwt 1 root   root          4096 Sep 13 12:27 .
drwxr-xr-x 1 root   root          4096 Sep 13 12:27 ..
-rw-r--r-- 1 root   root            55 Sep  7 01:34 .crates.toml
-rw-r--r-- 1 root   root           423 Sep  7 01:34 .crates2.json
drwxr-xr-x 2 hacker hacker        4096 Sep 13 12:27 .dojo
drwxr-xr-x 2 root   root          4096 Sep  7 01:34 bin
drwxr-xr-x 1 root   root          4096 Sep  7 01:23 hsperfdata_root
drwx------ 2 mysql  mysql         4096 Sep  7 01:24 tmp.TpSOPGOVKK
dr-xr-x--x 2 root   user_rakstfeu 4096 Sep 13 12:27 tmp333dzt0c
hacker@access-control~level11:~$ su user_rakstfeu
Password:
user_rakstfeu@access-control~level11:/home/hacker$ ls -la /tmp/tmp333dzt0c
total 12
dr-xr-x--x 2 root user_rakstfeu 4096 Sep 13 12:27 .
drwxrwxrwt 1 root root          4096 Sep 13 12:27 ..
-r--r----- 1 root user_knxbwkvf   60 Sep 13 12:27 tmp_ue9qi4f
user_rakstfeu@access-control~level11:/home/hacker$ cat /tmp/tmp333dzt0c/flag
cat: /tmp/tmp333dzt0c/flag: No such file or directory
```

Terminal | Flag

First screenshot continues into the next one.

```
In this challenge you will work understand how UNIX permissions for directories work with multiple users.

You'll be given access to various user accounts, use su to switch between them.


Created user user_rakstfeu with password wocsjuov
Created user user_knxbwkvf with password izumtiva
A copy of the flag has been placed somewhere in /tmp:
total 36
drwxrwxrwt 1 root   root          4096 Sep 13 12:27 .
drwxr-xr-x 1 root   root          4096 Sep 13 12:27 ..
-rw-r--r-- 1 root   root            55 Sep  7 01:34 .crates.toml
-rw-r--r-- 1 root   root           423 Sep  7 01:34 .crates2.json
drwxr-xr-x 2 hacker hacker        4096 Sep 13 12:27 .dojo
drwxr-xr-x 2 root   root          4096 Sep  7 01:34 bin
drwxr-xr-x 1 root   root          4096 Sep  7 01:23 hsperfdata_root
drwx------ 2 mysql  mysql         4096 Sep  7 01:24 tmp.TpSOPGOVKK
dr-xr-x--x 2 root   user_rakstfeu 4096 Sep 13 12:27 tmp333dzt0c
hacker@access-control~level11:~$ su user_rakstfeu
Password:
user_rakstfeu@access-control~level11:/home/hacker$ ls -la /tmp/tmp333dzt0c
total 12
dr-xr-x--x 2 root user_rakstfeu 4096 Sep 13 12:27 .
drwxrwxrwt 1 root root          4096 Sep 13 12:27 ..
-r--r----- 1 root user_knxbwkvf   60 Sep 13 12:27 tmp_ue9qi4f
user_rakstfeu@access-control~level11:/home/hacker$ cat /tmp/tmp333dzt0c/flag
cat: /tmp/tmp333dzt0c/flag: No such file or directory
user_rakstfeu@access-control~level11:/home/hacker$ su user_knxbwkvf
Password:
user_knxbwkvf@access-control~level11:/home/hacker$ cat /tmp/tmp333dzt0c/tmp_ue9qi4f
pwn.college{AOcPYFAFBU-b-2rx5I6RrWQ6XKX.QXyMDOzwCO2kjMzEzW}
user_knxbwkvf@access-control~level11:/home/hacker$
Terminal   Flag
```

Screenshot taken before successful capture.

```
In this challenge you will work understand how UNIX permissions for directories work with multiple users.

You'll be given access to various user accounts, use su to switch between them.


Created user user_rakstfeu with password wocsjuov
Created user user_knxbwkvf with password izumtiva
A copy of the flag has been placed somewhere in /tmp:
total 36
drwxrwxrwt 1 root   root          4096 Sep 13 12:27 .
drwxr-xr-x 1 root   root          4096 Sep 13 12:27 ..
-rw-r--r-- 1 root   root            55 Sep  7 01:34 .crates.toml
-rw-r--r-- 1 root   root           423 Sep  7 01:34 .crates2.json
drwxr-xr-x 2 hacker hacker        4096 Sep 13 12:27 .dojo
drwxr-xr-x 2 root   root          4096 Sep  7 01:34 bin
drwxr-xr-x 1 root   root          4096 Sep  7 01:23 hsperfdata_root
drwx------ 2 mysql  mysql         4096 Sep  7 01:24 tmp.TpSOPGOVKK
dr-xr-x--x 2 root   user_rakstfeu 4096 Sep 13 12:27 tmp333dzt0c
hacker@access-control~level11:~$ su user_rakstfeu
Password:
user_rakstfeu@access-control~level11:/home/hacker$ ls -la /tmp/tmp333dzt0c
total 12
dr-xr-x--x 2 root user_rakstfeu 4096 Sep 13 12:27 .
drwxrwxrwt 1 root root          4096 Sep 13 12:27 ..
-r--r----- 1 root user_knxbwkvf   60 Sep 13 12:27 tmp_ue9qi4f
user_rakstfeu@acc
cat: /tmp/tmp333
user_rakstfeu@acc
Password:
user_knxbwkvf@acc
pwn.college{AOcPYFAFBU-b-2rx5I6RrWQ6XKX.QXyMDOzwCO2kjMzEzW}
user_knxbwkvf@access-control~level11:/home/hacker$

🎉 Successfully completed level11! 🎉

Terminal   .QXyMDOzwCO2kjMzEzW}
```

After running the challenge, the flag was stored in the file **/tmp/tmp333dzt0c/tmp_ue9qi4f**. The file had permissions **-r--r-----**, meaning it was readable only by the **owner (root)** or the group **(user_knxbwkvf)**. I switched to **user_knxbwkvf** using the password **izumtiva** and then read the file directly with **cat /tmp/tmp333dzt0c/tmp_ue9qi4f** to retrieve the flag. This demonstrated how to access files by switching to a user account that has group read permissions, even when the directory permissions allow only specific users to list contents.

- **Level 12**

```
hacker@access-control~level12:~$ /challenge/run
===== Welcome to Access Control! =====
In this series of challenges, you will be working with various access control systems.
Break the system to get the flag.


In this challenge you will work understand how UNIX permissions for directories work with multiple users.

You'll be given access to various user accounts, use su to switch between them.


Created user user_xhznswfh with password hjyzfyic
Created user user_xchjjhbb with password lgqkzpzu
Created user user_bwmhvphx with password musvxvyn
A copy of the flag has been placed somewhere in /tmp:
total 36
drwxrwxrwt 1 root   root          4096 Sep 13 12:36 .
drwxr-xr-x 1 root   root          4096 Sep 13 12:35 ..
-rw-r--r-- 1 root   root            55 Sep  7 01:34 .crates.toml
-rw-r--r-- 1 root   root           423 Sep  7 01:34 .crates2.json
drwxr-xr-x 2 hacker hacker        4096 Sep 13 12:36 .dojo
drwxr-xr-x 2 root   root          4096 Sep  7 01:34 bin
drwxr-xr-x 1 root   root          4096 Sep  7 01:23 hsperfdata_root
drwx------ 2 mysql  mysql         4096 Sep  7 01:24 tmp.TpSOPGOVKK
dr-xr-x--x 3 root   user_xchjjhbb 4096 Sep 13 12:36 tmphcgwx6pr
hacker@access-control~level12:~$ su user_xchjjhbb
Password:

su: Authentication failure
hacker@access-control~level12:~$
hacker@access-control~level12:~$ su user_xchjjhbb
Password:
user_xchjjhbb@access-control~level12:/home/hacker$ ls -la /tmp/tmphcgwx6pr
```

First screenshot continues into the next one.

```
-rw-r--r-- 1 root   root            55 Sep  7 01:34 .crates.toml
-rw-r--r-- 1 root   root           423 Sep  7 01:34 .crates2.json
drwxr-xr-x 2 hacker hacker        4096 Sep 13 12:36 .dojo
drwxr-xr-x 2 root   root          4096 Sep  7 01:34 bin
drwxr-xr-x 1 root   root          4096 Sep  7 01:23 hsperfdata_root
drwx------ 2 mysql  mysql         4096 Sep  7 01:24 tmp.TpSOPGOVKK
dr-xr-x--x 3 root   user_xchjjhbb 4096 Sep 13 12:36 tmphcgwx6pr
hacker@access-control~level12:~$ su user_xchjjhbb
Password:

su: Authentication failure
hacker@access-control~level12:~$
hacker@access-control~level12:~$ su user_xchjjhbb
Password:
user_xchjjhbb@access-control~level12:/home/hacker$ ls -la /tmp/tmphcgwx6pr
total 12
dr-xr-x--x 3 root user_xchjjhbb 4096 Sep 13 12:36 .
drwxrwxrwt 1 root root          4096 Sep 13 12:36 ..
dr-xr-x--x 2 root user_bwmhvphx 4096 Sep 13 12:36 tmpksd0pea0
user_xchjjhbb@access-control~level12:/home/hacker$ su user_bwmhvphx
Password:
user_bwmhvphx@access-control~level12:/home/hacker$ cat /tmp/tmphcgwx6pr/<FLAGFILE>
bash: syntax error near unexpected token `newline'
user_bwmhvphx@access-control~level12:/home/hacker$ ls -la /tmp/tmphcgwx6pr/tmpksd0pea0
total 12
dr-xr-x--x 2 root user_bwmhvphx 4096 Sep 13 12:36 .
dr-xr-x--x 3 root user_xchjjhbb 4096 Sep 13 12:36 ..
-r--r----- 1 root user_xhznswfh   60 Sep 13 12:36 tmpm4txgs6q
user_bwmhvphx@access-control~level12:/home/hacker$ su user_xhznswfh
Password:
user_xhznswfh@access-control~level12:/home/hacker$ cat /tmp/tmphcgwx6pr/tmpksd0pea0/tmpm4txgs6q
pwn.college{wP7yMVoT56tU8ob-AbthSKN3Vyk.QXzMDOzwCO2kjMzEzW}
user_xhznswfh@access-control~level12:/home/hacker$
```

Screenshot taken before successful capture.

```
-rw-r--r-- 1 root   root            55 Sep  7 01:34 .crates.toml
-rw-r--r-- 1 root   root           423 Sep  7 01:34 .crates2.json
drwxr-xr-x 2 hacker hacker        4096 Sep 13 12:36 .dojo
drwxr-xr-x 2 root   root          4096 Sep  7 01:34 bin
drwxr-xr-x 1 root   root          4096 Sep  7 01:23 hsperfdata_root
drwx------ 2 mysql  mysql         4096 Sep  7 01:24 tmp.TpSOPGOVKK
dr-xr-x--x 3 root   user_xchjjhbb 4096 Sep 13 12:36 tmphcgwx6pr
hacker@access-control~level12:~$ su user_xchjjhbb
Password:

su: Authentication failure
hacker@access-control~level12:~$
hacker@access-control~level12:~$ su user_xchjjhbb
Password:
user_xchjjhbb@access-control~level12:/home/hacker$ ls -la /tmp/tmphcgwx6pr
total 12
dr-xr-x--x 3 root user_xchjjhbb 4096 Sep 13 12:36 .
drwxrwxrwt 1 root root          4096 Sep 13 12:36 ..
dr-xr-x--x 2 root user_bwmhvphx 4096 Sep 13 12:36 tmpksd0pea0
user_xchjjhbb@access-control~level12:/home/hacker$ su user_bwmhvphx
Password:
user_bwmhvphx@access-control~level12:/home/hacker$ cat /tmp/tmphcgwx6pr/<FLAGFILE>
bash: syntax error near unexpected token `newline'
user_bwmhvphx@access-control~level12:/home/hacker$ ls -la /tmp/tmphcgwx6pr/tmpksd0pea0
total 12
dr-xr-x--x 2 root user_bwmhvphx 4096 Sep 13 12:36 .
dr-xr-x--x 3 root
-r--r----- 1 root
user_bwmhvphx@acc
Password:
user_xhznswfh@acc
pwn.college{wP7yMVoT56tU8ob-AbthSKN3Vyk.QXzMDOzwCO2kjMzEzW}
user_xhznswfh@access-control~level12:/home/hacker$
```

🎉 Successfully completed **level12**! 🎉

Terminal :.QXzMDOzwCO2kjMzEzW}

After running the challenge, the flag was located in a nested directory structure within **/tmp**. I first switched to **user_xchjjhbb (password: lgqkzpzu)** to access the directory /tmp/tmphcgwx6pr. Then, I switched to **user_bwmhvphx (password: musvxvyn)** to access the **subdirectory tmpksd0pea0**. Finally, I switched to **user_xhznswfh (password: hjyzfyic)** to read the **file tmpm4txgs6q** because it was **group-readable** by **user_xhznswfh**. The flag was retrieved by reading the file with cat. This process involved switching users multiple times to leverage group permissions on directories and files, ultimately allowing access to the flag.