End-to-End File System Permissions Management in Windows Using PowerShell & ICACLS for Blue Team Operations

📍 Executive Summary

In this hands-on Windows lab, I engineered and executed a file permission management workflow using native PowerShell commands and the powerful `ICACLS` utility. The objective was to simulate real-world system administration and blue team scenarios where secure access control is critical.

Across multiple test cases, I granted, modified, and revoked file and folder permissions for specific users and groups — demonstrating the use of `ICACLS` to manage discretionary access control (DAC) in an enterprise Windows environment.

I also verified permission changes using both CLI and GUI methods, explained the implications of inherited vs. explicit permissions, and ensured alignment with best practices for hardening and role-based access control (RBAC).

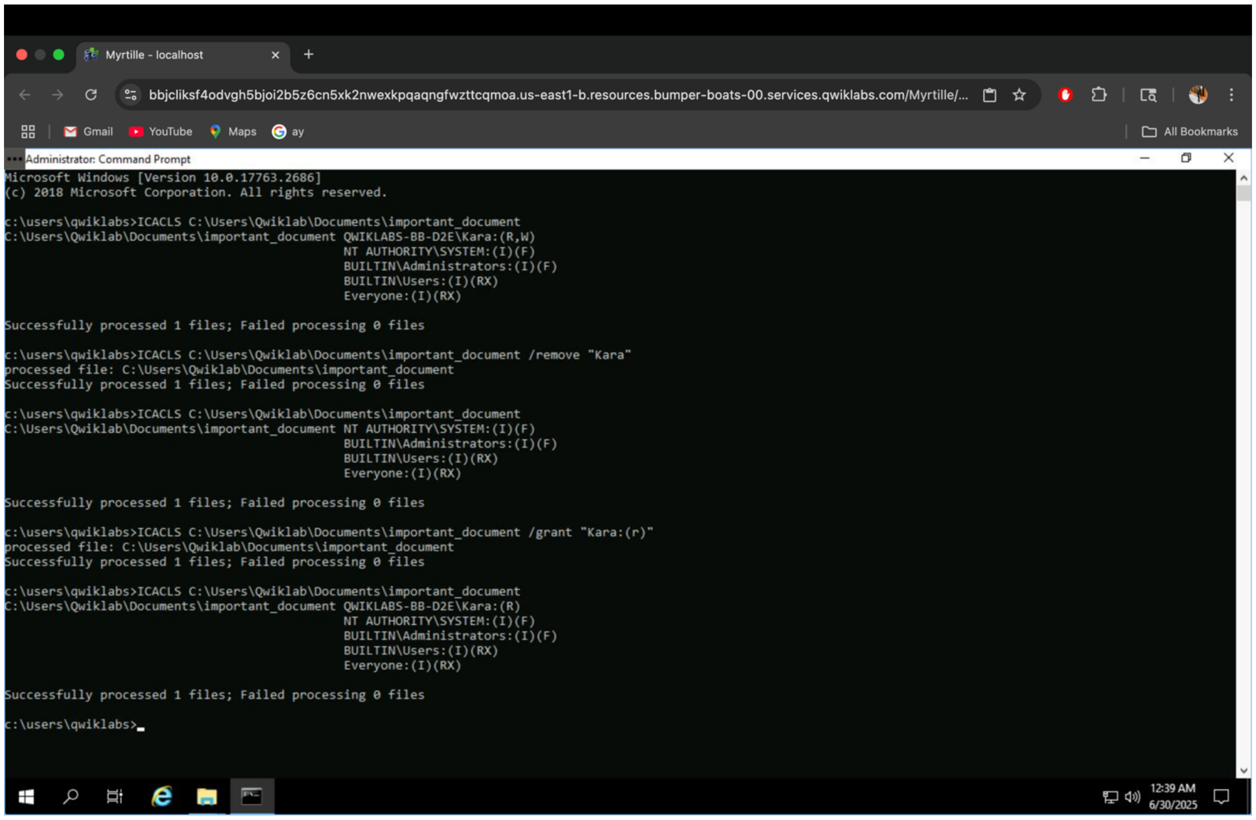
This lab directly maps to tasks frequently performed by Tier 1–2 SOC analysts, Windows sysadmins, and compliance engineers — especially during post-breach remediation, STIG compliance enforcement, or daily system hardening.

✅ Example 1: Restricting Write Access for a User

I had a file named `important_document` and noticed the user **Kara** had both **read and write** permissions. I wanted her to have **read-only** access.
So, I:

- Used `ICACLS` to view her current permissions.
- Removed Kara entirely from the file's ACL.
- Re-added her with **read-only** access using:
  `ICACLS C:\...\important_document /grant "Kara:(r)"`

This successfully enforced **least privilege**.


Myrtille - localhost
bbjcliksf4odvgh5bjoi2b5z6cn5xk2nwexkpqaqngfwzttcqmoa.us-east1-b.resources.bumper-boats-00.services.qwiklabs.com/Myrtille/...
Gmail
YouTube
Maps
ay
All Bookmarks
Administrator: Command Prompt
12:39 AM
6/30/2025

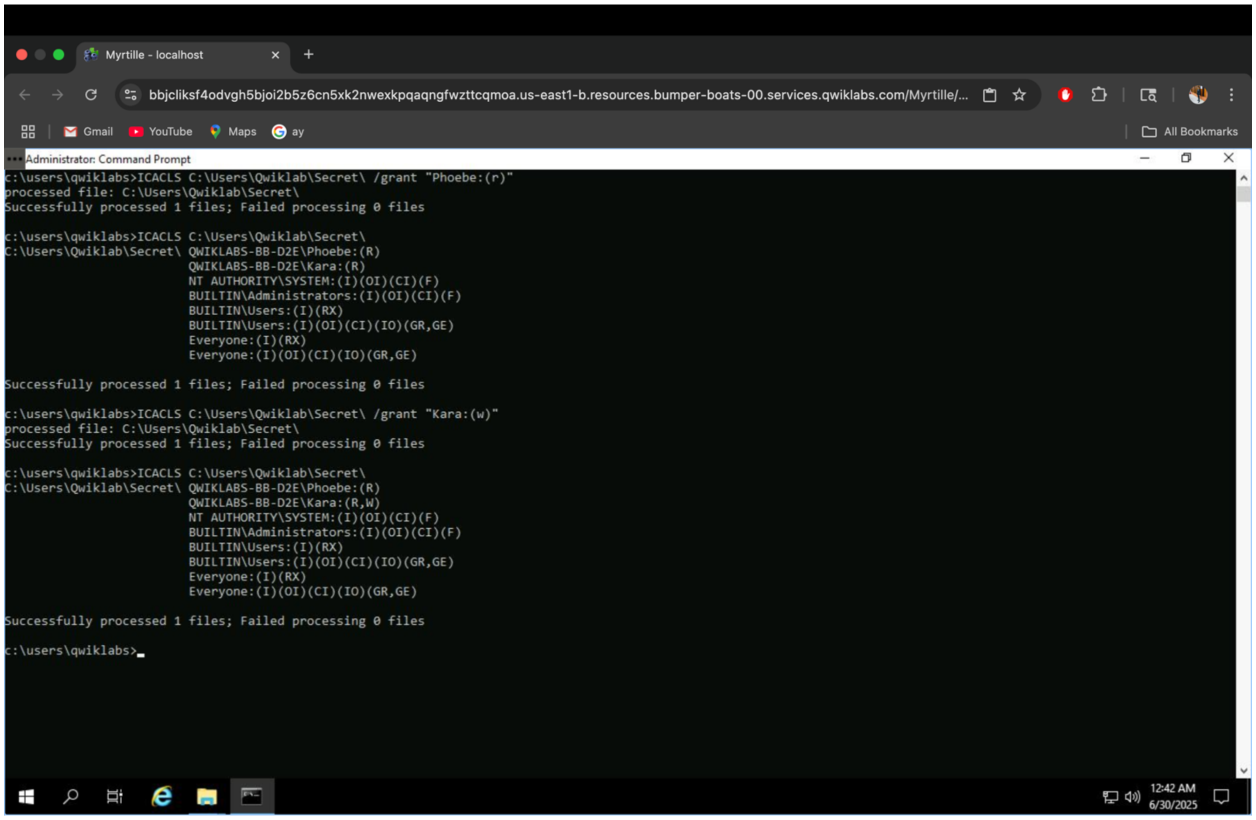
```
Microsoft Windows [Version 10.0.17763.2686]
(c) 2018 Microsoft Corporation. All rights reserved.

c:\users\qwiklabs>ICACLS C:\Users\Qwiklab\Documents\important_document
C:\Users\Qwiklab\Documents\important_document QWIKLABS-BB-D2E\Kara:(R,W)
                                              NT AUTHORITY\SYSTEM:(I)(F)
                                              BUILTIN\Administrators:(I)(F)
                                              BUILTIN\Users:(I)(RX)
                                              Everyone:(I)(RX)

Successfully processed 1 files; Failed processing 0 files

c:\users\qwiklabs>ICACLS C:\Users\Qwiklab\Documents\important_document /remove "Kara"
processed file: C:\Users\Qwiklab\Documents\important_document
Successfully processed 1 files; Failed processing 0 files

c:\users\qwiklabs>ICACLS C:\Users\Qwiklab\Documents\important_document
C:\Users\Qwiklab\Documents\important_document NT AUTHORITY\SYSTEM:(I)(F)
                                              BUILTIN\Administrators:(I)(F)
                                              BUILTIN\Users:(I)(RX)
                                              Everyone:(I)(RX)

Successfully processed 1 files; Failed processing 0 files

c:\users\qwiklabs>ICACLS C:\Users\Qwiklab\Documents\important_document /grant "Kara:(r)"
processed file: C:\Users\Qwiklab\Documents\important_document
Successfully processed 1 files; Failed processing 0 files

c:\users\qwiklabs>ICACLS C:\Users\Qwiklab\Documents\important_document
C:\Users\Qwiklab\Documents\important_document QWIKLABS-BB-D2E\Kara:(R)
                                              NT AUTHORITY\SYSTEM:(I)(F)
                                              BUILTIN\Administrators:(I)(F)
                                              BUILTIN\Users:(I)(RX)
                                              Everyone:(I)(RX)

Successfully processed 1 files; Failed processing 0 files

c:\users\qwiklabs>_
```

✅ Example 2: Managing Multiple Users' Access to a Folder

In the `Secret` folder:

- Kara had **read** access.
- I added **Phoebe** with **read** permission using:
  `ICACLS C:\...\Secret\ /grant "Phoebe:(r)"`
- Then gave Kara **write** access on top of her existing permissions:
  `ICACLS C:\...\Secret\ /grant "Kara:(w)"`

This helped me simulate how permissions evolve when users change roles.



Myrtille - localhost
bbjcliksf4odvgh5bjoi2b5z6cn5xk2nwexkpqaqngfwzttcqmoa.us-east1-b.resources.bumper-boats-00.services.qwiklabs.com/Myrtille/...
Gmail
YouTube
Maps
ay
All Bookmarks
Administrator: Command Prompt
12:42 AM
6/30/2025

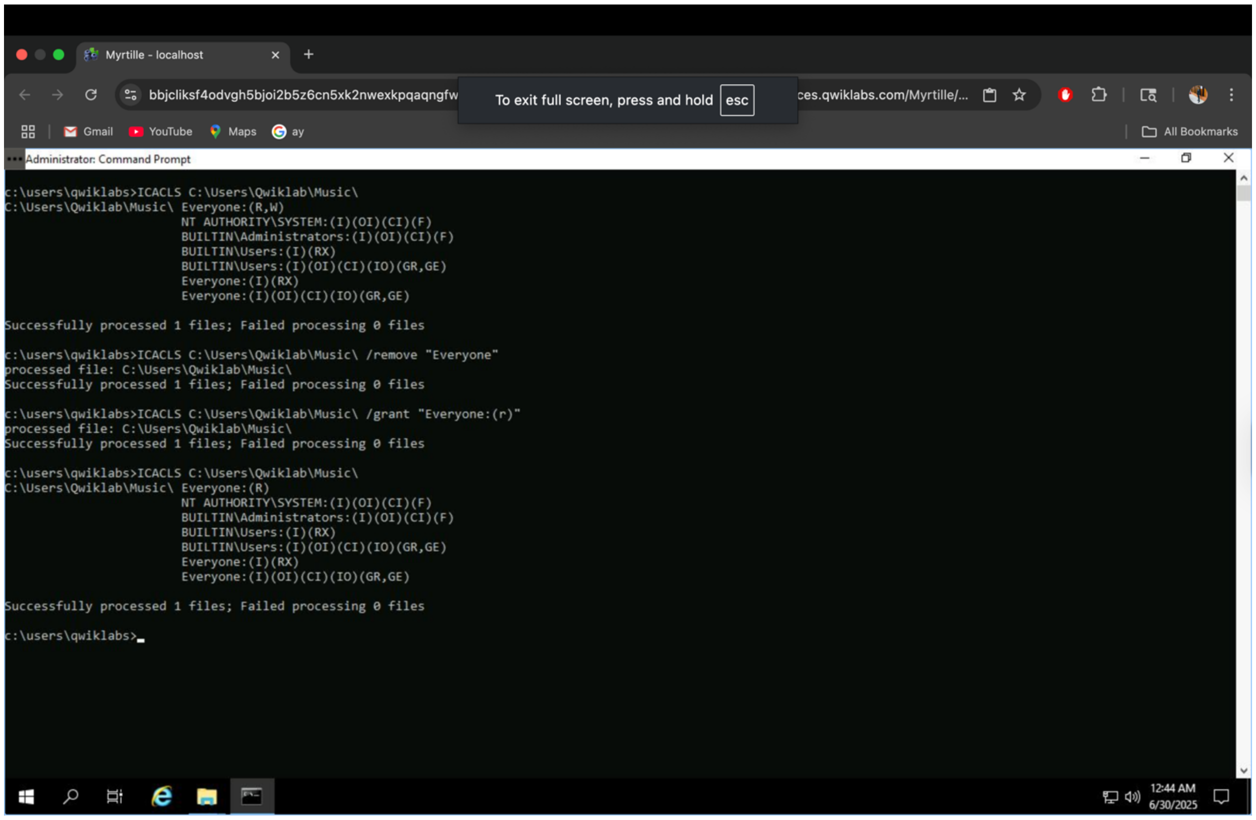
```
c:\users\qwiklabs>ICACLS C:\Users\Qwiklab\Secret\ /grant "Phoebe:(r)"
processed file: C:\Users\Qwiklab\Secret\
Successfully processed 1 files; Failed processing 0 files

c:\users\qwiklabs>ICACLS C:\Users\Qwiklab\Secret\
C:\Users\Qwiklab\Secret\ QWIKLABS-BB-D2E\Phoebe:(R)
                         QWIKLABS-BB-D2E\Kara:(R)
                         NT AUTHORITY\SYSTEM:(I)(OI)(CI)(F)
                         BUILTIN\Administrators:(I)(OI)(CI)(F)
                         BUILTIN\Users:(I)(RX)
                         BUILTIN\Users:(I)(OI)(CI)(IO)(GR,GE)
                         Everyone:(I)(RX)
                         Everyone:(I)(OI)(CI)(IO)(GR,GE)

Successfully processed 1 files; Failed processing 0 files

c:\users\qwiklabs>ICACLS C:\Users\Qwiklab\Secret\ /grant "Kara:(w)"
processed file: C:\Users\Qwiklab\Secret\
Successfully processed 1 files; Failed processing 0 files

c:\users\qwiklabs>ICACLS C:\Users\Qwiklab\Secret\
C:\Users\Qwiklab\Secret\ QWIKLABS-BB-D2E\Phoebe:(R)
                         QWIKLABS-BB-D2E\Kara:(R,W)
                         NT AUTHORITY\SYSTEM:(I)(OI)(CI)(F)
                         BUILTIN\Administrators:(I)(OI)(CI)(F)
                         BUILTIN\Users:(I)(RX)
                         BUILTIN\Users:(I)(OI)(CI)(IO)(GR,GE)
                         Everyone:(I)(RX)
                         Everyone:(I)(OI)(CI)(IO)(GR,GE)

Successfully processed 1 files; Failed processing 0 files

c:\users\qwiklabs>_
```

✅ Example 3: Restricting Group Access for "Everyone"

In the `Music` folder, the **Everyone** group had **read and write** access. That's way too open. So, I:

- Removed the group completely using `/remove`.
- Re-granted **read-only** access with `/grant "Everyone:(r)"`.

This was a great real-world exercise in **tightening overly permissive group access**.

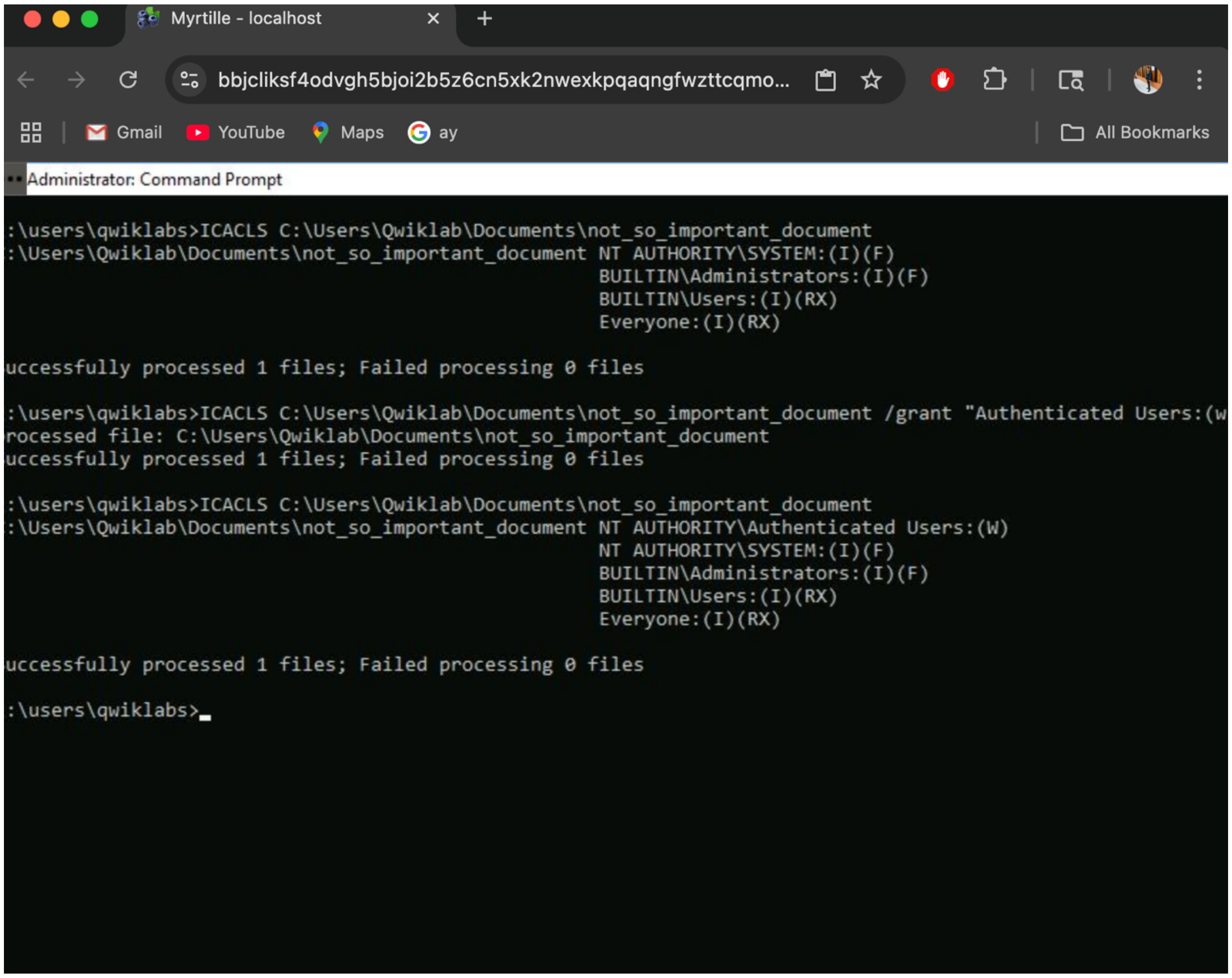

Myrtille - localhost
To exit full screen, press and hold esc
Administrator: Command Prompt
12:44 AM
6/30/2025

```
c:\users\qwiklabs>ICACLS C:\Users\Qwiklab\Music\
C:\Users\Qwiklab\Music\ Everyone:(R,W)
                        NT AUTHORITY\SYSTEM:(I)(OI)(CI)(F)
                        BUILTIN\Administrators:(I)(OI)(CI)(F)
                        BUILTIN\Users:(I)(RX)
                        BUILTIN\Users:(I)(OI)(CI)(IO)(GR,GE)
                        Everyone:(I)(RX)
                        Everyone:(I)(OI)(CI)(IO)(GR,GE)

Successfully processed 1 files; Failed processing 0 files

c:\users\qwiklabs>ICACLS C:\Users\Qwiklab\Music\ /remove "Everyone"
processed file: C:\Users\Qwiklab\Music\
Successfully processed 1 files; Failed processing 0 files

c:\users\qwiklabs>ICACLS C:\Users\Qwiklab\Music\ /grant "Everyone:(r)"
processed file: C:\Users\Qwiklab\Music\
Successfully processed 1 files; Failed processing 0 files

c:\users\qwiklabs>ICACLS C:\Users\Qwiklab\Music\
C:\Users\Qwiklab\Music\ Everyone:(R)
                        NT AUTHORITY\SYSTEM:(I)(OI)(CI)(F)
                        BUILTIN\Administrators:(I)(OI)(CI)(F)
                        BUILTIN\Users:(I)(RX)
                        BUILTIN\Users:(I)(OI)(CI)(IO)(GR,GE)
                        Everyone:(I)(RX)
                        Everyone:(I)(OI)(CI)(IO)(GR,GE)

Successfully processed 1 files; Failed processing 0 files

c:\users\qwiklabs>_
```

✅ Example 4: Granting Write Access to "Authenticated Users"

For the `not_so_important_document`, **Authenticated Users** weren't listed at all.
I added them with write access:
`ICACLS C:\...\not_so_important_document /grant "Authenticated Users:(w)"`

This simulated giving domain-authenticated users write rights without over-permissioning others.


Myrtille - localhost
bbjcliksf4odvgh5bjoi2b5z6cn5xk2nwexkpqaqngfwzttcqmo...
Gmail
YouTube
Maps
ay
All Bookmarks
Administrator: Command Prompt

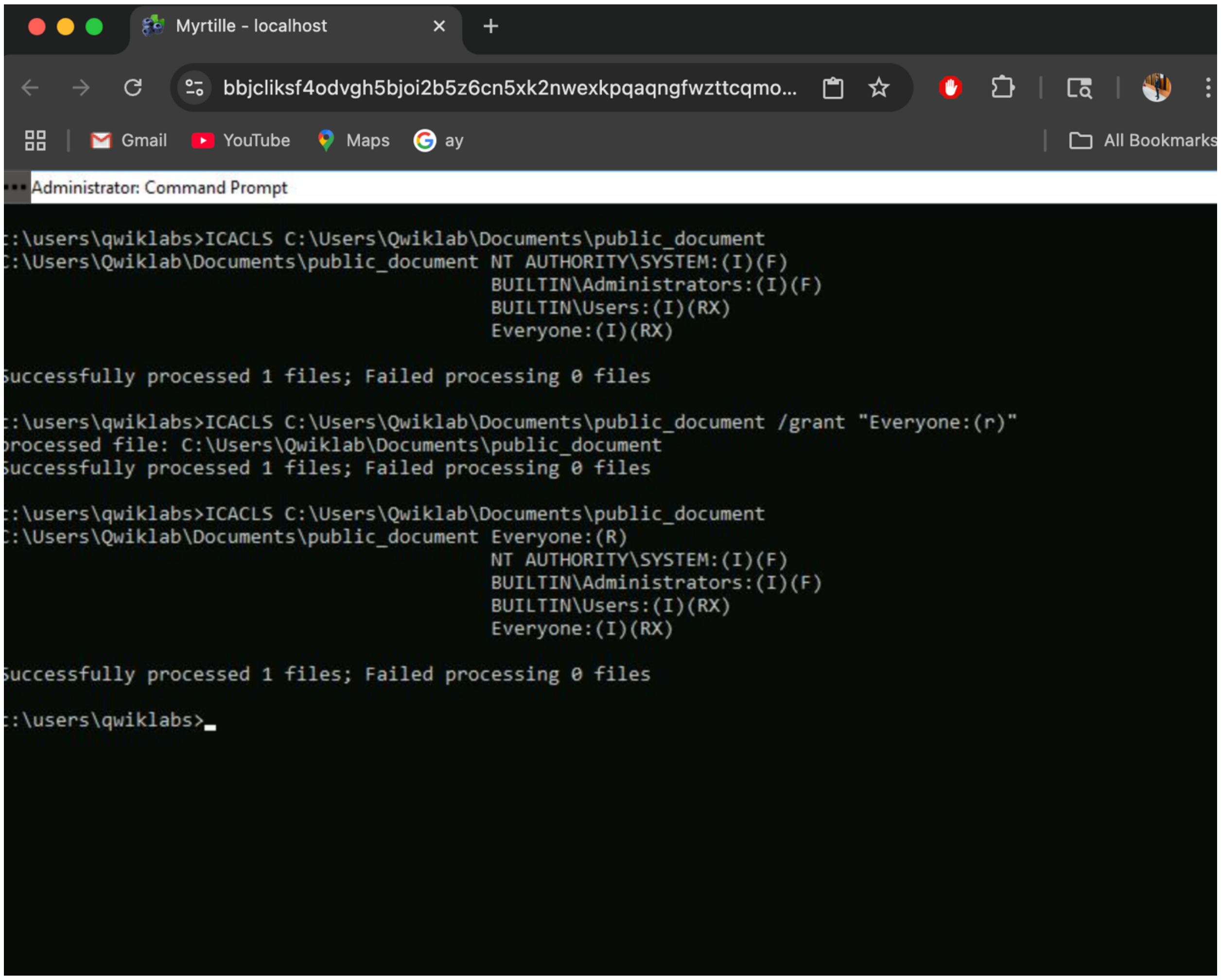
```
:\users\qwiklabs>ICACLS C:\Users\Qwiklab\Documents\not_so_important_document
:\Users\Qwiklab\Documents\not_so_important_document NT AUTHORITY\SYSTEM:(I)(F)
                                                    BUILTIN\Administrators:(I)(F)
                                                    BUILTIN\Users:(I)(RX)
                                                    Everyone:(I)(RX)

uccessfully processed 1 files; Failed processing 0 files

:\users\qwiklabs>ICACLS C:\Users\Qwiklab\Documents\not_so_important_document /grant "Authenticated Users:(w
rocessed file: C:\Users\Qwiklab\Documents\not_so_important_document
uccessfully processed 1 files; Failed processing 0 files

:\users\qwiklabs>ICACLS C:\Users\Qwiklab\Documents\not_so_important_document
:\Users\Qwiklab\Documents\not_so_important_document NT AUTHORITY\Authenticated Users:(W)
                                                    NT AUTHORITY\SYSTEM:(I)(F)
                                                    BUILTIN\Administrators:(I)(F)
                                                    BUILTIN\Users:(I)(RX)
                                                    Everyone:(I)(RX)

uccessfully processed 1 files; Failed processing 0 files

:\users\qwiklabs>_
```

✅ Example 5: Making a File Publicly Readable

I wanted `public_document` to be readable by **any user** on the system.
Rather than adding each user, I granted **Everyone** read access:

```
ICACLS C:\...\public_document /grant "Everyone:(r)"
```

Super useful for setting up shared/public files while maintaining control.


Myrtille - localhost
bbjcliksf4odvgh5bjoi2b5z6cn5xk2nwexkpqaqngfwzttcqmo...
Gmail
YouTube
Maps
ay
All Bookmarks
Administrator: Command Prompt

```
:\users\qwiklabs>ICACLS C:\Users\Qwiklab\Documents\public_document
:\Users\Qwiklab\Documents\public_document NT AUTHORITY\SYSTEM:(I)(F)
                                          BUILTIN\Administrators:(I)(F)
                                          BUILTIN\Users:(I)(RX)
                                          Everyone:(I)(RX)

uccessfully processed 1 files; Failed processing 0 files

:\users\qwiklabs>ICACLS C:\Users\Qwiklab\Documents\public_document /grant "Everyone:(r)"
rocessed file: C:\Users\Qwiklab\Documents\public_document
uccessfully processed 1 files; Failed processing 0 files

:\users\qwiklabs>ICACLS C:\Users\Qwiklab\Documents\public_document
:\Users\Qwiklab\Documents\public_document Everyone:(R)
                                          NT AUTHORITY\SYSTEM:(I)(F)
                                          BUILTIN\Administrators:(I)(F)
                                          BUILTIN\Users:(I)(RX)
                                          Everyone:(I)(RX)

uccessfully processed 1 files; Failed processing 0 files

:\users\qwiklabs>_
```

✅ Conclusion

This lab reinforced essential skills in file system security, Windows access control, and PowerShell scripting — all of which are foundational for blue teamers in SOC environments.

By mastering `ICACLS`, I demonstrated how to:

- Apply granular user/group permissions on files and folders
- Troubleshoot broken ACLs and resolve inheritance issues
- Use command-line tools to audit and enforce access policies without relying on the GUI

These techniques are directly applicable to real-world cybersecurity operations — including compliance hardening, insider threat mitigation, and lateral movement prevention.

As organizations increasingly enforce least privilege and RBAC policies, the ability to efficiently manage permissions through PowerShell is a must-have skill for any aspiring SOC analyst or Windows security specialist. 💪🔐