File and Registry Permissions , Lockdown and difference between user, admin and system context

GROUP C

File Permissions in Application Packaging

- Control who can read, write, modify, or execute files.
- Common Scenarios:
- • Writable application data folders (e.g., %ProgramData%, %AppData%)
- • Installation folders usually read/execute for users, full control for admins
- • Logs and temp files may need modify access for normal users

Types of Permissions

- **File Permissions (NTFS**)
- Read – View content.
- Write – Add or modify content.
- Execute – Run programs.
- Modify – Read, write, delete.
- Full Control – All actions, including permission changes.

Why we set registry Permission?

- In Application Packaging, registry permissions simply mean deciding who can open, change, or delete a registry setting that belongs to the application-
- To make sure only the right people can change important settings.
- To let the app save changes (like user preferences) without giving too much access.
- Simple example
- License key in the registry → only Admins can change it, users can just read it.
- User settings in the registry → give users permission to write, so the app can save their changes.

Registry Permission Levels

- Full Control – Create, delete, modify keys/values
- Read – View keys and values
- Write – Add or change values

Types of Registry Permission

- 1. Read
- Can only see the registry values.
- Example: Users can read a license key but can't change it.
- 2. Write
- Can add or change values in the registry.
- Example: App saves user preferences like theme or language.
- 3. Full Control
- Can read, write, delete, and change permissions.
- Example: Usually given only to Admins, not normal users.
- 4. Special Permissions
- Custom access (a mix of read, write, or other rights).
- Example: Allow users to add values but not delete them.

Lockdown

- **What it means?**
- A lockdown is when you limit or block access to certain features, files, or settings of an application so that users can't change, break, or misuse it.
- This is often done via permissions, Group Policy Objects (GPO), registry changes, or application configuration settings during packaging.

Why lockdown is done?

- Prevent accidental or malicious changes.
- Maintain the app in a tested, working state.
- Protect sensitive data and system integrity.

Tools used for implementing Lockdown in Application packaging

- 1. MSI Editors – change the app setup to remove or disable features.
- InstallShield
- AdminStudio
- ORCA
- 2. Permissions & Policies – control what users can access
- Group Policy (GPO)
- NTFS folder/file permissions

Tools used for implementing Lockdown in Application packaging

- 3. Scripting Tools – add custom restrictions during install
- PowerShellVBScript / Batch files
- 4. Registry & Config Changes – turn off features through settings
- Registry Editor (regedit)
- App config files (INI/XML)
- 5. App Control Tools – block unapproved apps from running
- Microsoft AppLocker
- Ivanti Application Control

User

- The program can only do what you, as a normal user, are allowed to do.
- It can open your personal files, make changes in your own folders (like Documents or Downloads), and change settings that only affect your account.
- It cannot change system-wide settings, install software for all users, or access other users' private files.
- If it tries to do something "big" (like changing Windows settings), Windows will ask for an Admin password or show a permission denied message.
- Example : Opening Notepad and editing a text file saved on your Desktop — that's in user context.

Admin

- When you run something in admin context, you're telling the computer:
- "Hey, I'm the boss here. Let me do whatever I want on this system."
- It's like having master keys to a building — you can open any door, change the furniture, or even replace the locks.

How it works in Windows

- Your computer normally protects certain areas (like C:\Windows or system registry keys).
- If a program wants to change those, Windows pops up the UAC (User Account Control) box asking:“Do you allow this app to make changes?”
- If you click Yes (and you have admin rights), the program runs in Admin Context.

Real-Life Examples

- 1. Installing Microsoft Office
- 2. Changing Wi-Fi Settings for Everyone
- 3. Removing Built-in Windows Apps

System

- Represents the **application itself or automated background processes** acting independently of any specific human user.
- **Scope & Permissions**
- Full, unrestricted access to **all data and system functions**
- Can perform actions on behalf of users or administrators
- Runs scheduled or event-driven processes automatically

System

- **Purpose**
- Keep the application running smoothly
- Perform repetitive or time-based maintenance
- Integrate with other services and APIs
- **Examples**
- Automatically delete spam or inactive accounts
- Send scheduled email reminders or reports
- Process payments through an external payment gateway
- Generate daily database backups

System

- **Risks**
- **Highest security risk** — if the system context is compromised, attackers gain total control
- Needs strict API key security, logging, and monitoring

Difference between user, admin and system.

Feature	User Context	Admin Context	System Context
Who	Logged-in end user	Privileged human user	Application or automation
Scope	Own data only	All users' data and settings	Entire system & all data
Purpose	Personal usage	Manage users & system	Maintain & operate system
Example	Edit own blog	Delete any blog	Auto-delete spam
Permission Level	Low	High	Full/unlimited
Human Interaction	Yes	Yes	No (runs automatically)
Security Risk	Low	Medium-High	Very High