1. **BUGS**. Software bugs are an error or failure in software and they're very common. Some bugs will result in serious issues like information theft and soma will lead to system failure. But some less serious bugs will result in error messages or incorrect results. Bugs, in general, cause the software to behave in an unexpected manner. Pretty much all software contains minor (or major) bugs. Hackers can easily take advantage of soma software bugs and cause much harm if you do not fix security vulnerabilities. While it's usually impossible to ship software with 0 bugs, it's important to find and fix any serious bugs, especially ones that could pose a security risk.

2. **EXPOSURE OF SENSITIVE DATA**. Sensitive data includes things such as account numbers, addresses, financial data, health information, usernames, and passwords. All this data must be protected to keep it from falling into wrong hands. Personal or sensitive data has to be protected with encryption and access controls to prevent unauthorized people from accessing it. If the software fails to protect this personal data due to security vulnerabilities, hackers who gain access to this information can use it to commit fraud and other crimes.

3. **MALWARE RECOVERY**.A backup system must be able to create restore points that allow an administrator to restore the system to some point in time. A traditional backup system accomplishes this by coping the data to another storage device, while a snapshot system uses multiple snapshots to create virtual views of the data set at different points in time.
Administrators often use snapshots to restore files that have been modified by malware to their original state. This capability is especially useful for ransomware, which prevents users from accessing content until a ransom has been paid. Advanced ransomware can change a file's name, which can prevent traditional backup system from recovering the file. However, a snapshot could restore the file to its state before it was infected by ransomware.

4. **DISASTER RECOVERY**. Disaster recovery is another common use to backup systems. The traditional method involves creating a backup set on portable media such as tapes and sending it to an off-site storage facility. A snapshot-based backup accomplishes this task with replication, which allows the backup system to place copies of the data in multiple locations. This scenario may involve using the same replication stream to send a backup set to both an on-site and off-site storage system. The on-site backup may be used for operational recovery, while the off-site backup is used to disaster recovery. A system administrator could accomplish this by replicating the primary storage directly to both backup storage sites. Another approach is to replicating the data to the on site storage first, then replicate the data from the on-site storage to the off-site storage.

5. **TESTING**. Snapshots make testing a lot easier. A snapshot backup can easily be cloned and create another VM that has the exact configuration/data as a production VM. From this cloned VM, code changes or operation system upgrades can be tested before making those changes to a production environment. If upgrades are made to a production VM and they don't go well, it is fast and easily to rollback the changes to a previous snapshot.

6. **SNAPSHOT BACKUPS**. The Storage Networking Industry Association (SNIA) defines a backup as a collection of data stored on non-volatile storage media for the purpose of recovery. According to this definition, a snapshots isn't a backup itself until the data has been replicated to storage device because a snapshot is a virtual copy of the data, rather than an actual copy of the data. Snapshots provide a means of performing backups significantly more quickly than traditional method, especially for large data sets. The time required for traditional backup is

proportional to the data set, whereas the time needed for a snapshot doesn't increase with the size of the data set. Some snapshot implementations take an initial snapshot of the entire data set and only use subsequent snapshots for changed data by referencing unchanged data with a set of pointers to the initial snapshot. This approach also use less storage space than repeatedly cloning unchanged data. Snapshots also avoid the problem of version skewing, which allows application to continue writing data.

7. **TRADITIONAL BACKUPS.** A traditional backup system places files in some type of compressed format to minimize the size of the backup set. It then stores the backup set on another storage medium, often tape or disk. The process of compressing the files and copying then makes a full backup a time-consuming process. This is especially true in the case of a multi-tasking system that may create changes to the files as they're especially true in the case of a multi-tasking system that may create changes to the files as they're being backed up, which results in the backup having a condition known as version skew. Assume for this example of version skew that a file is moved into a directory after it ha already been backed up but before the backup is complete. The file won't exist at all on the backup, even though it was created before the backup finished. Version skew is especially problematic when a file's contents are changed while the backup program is reading it, which can corrupt the file. One approach to avoiding version skew when backing up live data is to disable write operations to the disk during the backup. This typically requires the administrator to terminate the applications that could write to the disk or use an Application Programmer Interface (API) to deny write access. This procedure may be acceptable for systems such as a desktop computer that can tolerate regular downtime, but servers that require a higher availability can't bear a service outage every time a backup is performed.

8. **BROKEN/MISSING AUTHENTICATION.** Weaknesses in session management and credential management result in broken authentication, which means an attacker is able to compromise passwords or other information to access a user's account. Improperly implemented authentication and session management can result in this kind of software vulnerability. There are a lot of adverse effects that can occur as a result of software security weaknesses. But you can prevent these problems if you take all the necessary security precautions while developing the software. It's important for software developers to use different methods to detected weaknesses in their software automatically.