Validate All Input

Always validate user input by testing type, length, format, and range. When you are implementing precautions against malicious input, consider the architecture and deployment scenarios of your application. Remember that programs designed to run in a secure environment can be copied to an nonsecure environment. The following suggestions should be considered best practices:

- Make no assumptions about the size, type, or content of the data that is received by your application. For example, you should make the following evaluation:
    - How will your application behave if an errant or malicious user enters a 10-megabyte MPEG file where your application expects a postal code?
    - How will your application behave if a DROP TABLE statement is embedded in a text field?

- Test the size and data type of input and enforce appropriate limits. This can help prevent deliberate buffer overruns.

- Test the content of string variables and accept only expected values. Reject entries that contain binary data, escape sequences, and comment characters. This can help prevent script injection and can protect against some buffer overrun exploits.

- When you are working with XML documents, validate all data against its schema as it is entered.

- Never build Transact-SQL statements directly from user input.

- Use stored procedures to validate user input.

- In multitiered environments, all data should be validated before admission to the trusted zone. Data that does not pass the validation process should be rejected and an error should be returned to the previous tier.

- Implement multiple layers of validation. Precautions you take against casually malicious users may be ineffective against determined attackers. A better practice is to validate input in the user interface and at all subsequent points where it crosses a trust boundary.
  For example, data validation in a client-side application can prevent simple script injection. However, if the next tier assumes that its input has already been validated, any malicious user who can bypass a client can have unrestricted access to a system.

- Never concatenate user input that is not validated. String concatenation is the primary point of entry for script injection.

- Do not accept the following strings in fields from which file names can be constructed: AUX, CLOCK$, COM1 through COM8, CON, CONFIG$, LPT1 through LPT8, NUL, and PRN.

When you can, reject input that contains the following characters.

;	Query delimiter.
'	Character data string delimiter.
--	Comment delimiter.
/* ... */	Comment delimiters. Text between /* and */ is not evaluated by the server.
xp_	Used at the start of the name of catalog-extended stored procedures, such as **xp_cmdshell**.