Secure Coding Practices

TCS 591 : UNIT 1

Top 10
Secure Coding Practices from the Computer Emergency Response Team (CERT)

1. **Validate input**.
2. **Heed compiler warnings.**
3. **Architect and design for security policies.**
4. **Keep it simple.**
5. **Default to deny.**
6. **Adhere to the principle of least privilege.**
7. **Sanitize data sent to other systems.**
8. **Practice defence in depth.**
9. **Use effective quality-assurance techniques.**
10. **Adopt a secure coding standard.**

Top 10
Secure Coding Practices

- **Validate input.** Validate input from all untrusted data sources. Proper input validation can eliminate the vast majority of software vulnerabilities. Be suspicious of most external data sources, including command line arguments, network interfaces, environmental variables, and user controlled files
- **Heed compiler warnings.** Compile code using the highest warning level available for your compiler and eliminate warnings by modifying the code .Use static and dynamic analysis tools to detect and eliminate additional security flaws.

Top 10 Secure Coding Practices

- **Architect and design for security policies.** Create a software architecture and design your software to implement and enforce security policies. For example, if your system requires different privileges at different times, consider dividing the system into distinct intercommunicating subsystems, each with an appropriate privilege set.
- **Keep it simple.** Keep the design as simple and small as possible .Complex designs increase the likelihood that errors will be made in their implementation, configuration, and use. Additionally, the effort required to achieve an appropriate level of assurance increases dramatically as security mechanisms become more complex.

Top 10
Secure Coding Practices

- **Default deny.** Base access decisions on permission rather than exclusion. This means that, by default, access is denied and the protection scheme identifies conditions under which access is permitted
- **Adhere to the principle of least privilege.** Every process should execute with the the least set of privileges necessary to complete the job. Any elevated permission should only be accessed for the **least amount of time** required to complete the privileged task. This approach reduces the opportunities an attacker has to execute arbitrary code with elevated privileges

Top 10
Secure Coding Practices

- **Sanitize data sent to other systems.** Sanitize all data passed to complex subsystems such as command shells, relational databases, and commercial off-the-shelf (COTS) components. Attackers may be able to invoke unused functionality in these components through the use of SQL, command, or other injection attacks. This is not necessarily an input validation problem because the complex subsystem being invoked does not understand the context in which the call is made. Because the calling process understands the context, it is responsible for sanitizing the data before invoking the subsystem.

Top 10
Secure Coding Practices

- **Practice defense in depth.** Manage risk with multiple defensive strategies, so that if one layer of defence turns out to be inadequate, another layer of defense can prevent a security flaw from becoming an exploitable vulnerability and/or limit the consequences of a successful exploit. For example, combining secure programming techniques with secure runtime environments should reduce the likelihood that vulnerabilities remaining in the code at deployment time can be exploited in the operational environment
- **Use effective quality assurance techniques.** Good quality assurance techniques can be effective in identifying and eliminating vulnerabilities. Fuzz testing, penetration testing, and source code audits should all be incorporated as part of an effective quality assurance program. Independent security reviews can lead to more secure systems. External reviewers bring an independent perspective; for example, in identifying and correcting invalid assumptions
- **Adopt a secure coding standard.** Develop and/or apply a secure coding standard for your target development language and platform.