Tecniche per lo sviluppo di codice sicuro

Francesco Pugliese, PhD

neural1977@gmail.com

Tecniche per lo sviluppo di codice sicuro

- ✓ **L'analisi di Sicurezza** rappresenta il primo passo per realizzare un prodotto che soddisfi i principi di **confidenzialità**, **integrità**, e **disponibilità**. Ecco come procedere allo sviluppo di software che siano stabili e sicuri.

- ✓ Lo **Sviluppo del Software** è il **mix** di design, architettura, programmazione e test.

- ✓ Non avere la **sicurezza** come requisito porta a un prodotto **scadente**, perdite di tempo e **costi** inutili.

Tecniche per lo sviluppo di codice sicuro

- **Lo sviluppo di un software** – tutto ciò che contiene del codice di **programmazione**, come un eseguibile per computer, sito web, DBMS o applicazione mobile – è una attività **onerosa** che richiede tempo e **investimenti economici**.

- Le **attività di sviluppo** sono svolte da professionisti del **design**, **progettazione**, **programmazione** e **test**, generalmente con notevoli competenze nei relativi settori. La fase di **controllo** della **sicurezza** – **confidenzialità**, **integrità**, **disponibilità** – è però troppo spesso ignorata o eseguita con **superficialità**.

Secure Coding

- Il non lavorare secondo le regole e linee guida del **Secure Coding** può portare a ottenere un prodotto **scadente**, e quindi a perdere la **fiducia di clienti**, fornitori e di tutti gli altri **stakeholder**, oltre a importanti **danni economici** dovuti a blocchi nelle attività, **perdita di dati** o ad **attacchi informatici**.

- Scoperta la **problematica**, sarà necessaria poi un'attività di **analisi profonda** – per comprendere **l'errore**, quali sono i rischi e le risorse coinvolte – seguita dalla fase di test per **validare** i risultati ottenuti dall'analisi e da una fase di **aggiornamento** o **riparazione** del codice, per poi effettuare nuovamente analisi e test per verificare il **miglioramento apportato**

Secure Coding

- La **mancanza** di un approccio **proattivo** è la prima causa di **vulnerabilità** in un software, la maggior parte derivanti da un numero relativamente piccolo di errori comuni di programmazione. Queste vulnerabilità sono note e largamente documentate su Internet: ciò le rende facilmente sfruttabili da un malintenzionato.

- Prevedere la **sicurezza** nel codice permette quindi di evitare l'introduzione accidentale di **vulnerabilità**, bug, e malfunzionamenti **funzionali** e/o logici.

- In base alla natura del **software**, dell'infrastruttura su cui opera e della vulnerabilità, gli impatti possono **compromettere** il software, i sistemi operativi, i database, l'ambiente condiviso o anche il sistema dell'utente/cliente, e tutte le informazioni **associate**.

Vulnerabilità Note

- ✓ Individuiamo alcune tra le **vulnerabilità** più conosciute. Gli esempi riportati non sono i problemi più **gravi**, ma i più comunemente commessi.

- ✓ **Overflow:** Il **Buffer Overflow** si verifica quando un **processo** tenta di archiviare dati oltre il **limite fisso prestabilito.** Se ad esempio è possibile memorizzare solo 10 elementi, i successivi saranno scritti ripartendo dall'inizio della memoria dedicata. In base a come viene gestita la **problematica** di default, il sistema può bloccarsi, interrompere la scrittura dei nuovi dati, o (solitamente) riscrivere sui vecchi dati.

- ✓ Un **Integer Overflow** si verifica quando un'operazione aritmetica genera un numero troppo grande per essere rappresentato all'interno dello **spazio disponibile**.

Bibliografia

https://www.cybersecurity360.it/cybersecurity-nazionale/secure-coding-regole-e-linee-guida-per-lo-sviluppo-software-sicuro/