Secure code

Jacopo De Angelis

16 luglio 2020

Indice

Capitolo 1

Concetti di sicurezza

1.1 Privilegi minimi

Nel caso i permessi non siano dati correttamente, un utente malintenzionato potrebbe decidere di accedere ai dati in maniera distruttiva, ad esempio tramite una SQL injection.

In questo caso una soluzione è cambiare i permessi per i vari gruppi in modo che abbiano il potere minimo per le loro funzioni. Ad esempio, nel caso di un database di un sito dove le query non sono sanificate (ovvero è possibile eseguire codice di accesso al DB all'interno delle query), si può decidere che gli utenti abbiano solo possibilità di lettura e non di scrittura.

Nel caso l'utente malintenzionato acceda tramite un ruolo da amministratore potrebbe decidere di cambiare password dei database, cambiare dati o interrompere il servizio. In più qualsiasi codice eseguito da un utente con certi privilegi verrebbe eseguito con gli stessi, causando altri tipi di malfunzionamento

Il concetto dei privilegi minimi non corregge le vulnerabilità del sistema, come le code injection, ma rende più difficile per un utente malintenzionato sfruttare queste falle

In generale i processi avviati dall'applicazione dovrebbero essere eseguiti con i privilegi minimi necessari

1.2 Difesa in profondità

Il concetto dietro alla difesa in profondità è quello di creare più livelli di sicurezza, di diverso tipo, tramite diversi sistemi. Ad esempio:

- Dividere i server in più network in modo che l'attacco non possa propagarsi
- inserire un firewall
- criptare i dati del database
- sanificare le query e parametrizzarle
- usare validazione sia a livello di frontend che di backend
- mettere in sicurezza i dati nel mondo fisico, ad esempio tramite guardie ai server, accesso solo tramite badge, ecc.

Implementare sempre più livelli di sicurezza, di più tipi, fisici, architetturali e di codice

Livello dati: controllo degli accessi, criptazione, backup

Livello applicazione: autenticazione, autorizzazione, auditing (noti assieme come AAA), codice sicuro e "hardening"

Livello host: "hardening", autenticazione, gestione degli aggiornamenti, antivirus

Livello di network interno: Segmentazione del network, IPsec, TLS, NAT

Livello perimetrale: firewall, TLS, negazione dell'accesso, prevenzione

Sicurezza fisica: guardie, lucchetti, dispositivi di tracciamento, badge

1.3 Sicuro per default

Il concetto dietro ad un software sicuro per default è che le impostazioni di sicurezza siano correttamente impostate.

Per fare un esempio, la richiesta affinchè la password inserita sia sicura (lunghezza minima, caratteri speciali, ecc.) è una buona misura di sicurezza che dovrebbe essere implementata di base. In questo modo gli utenti sarebbero già più al sicuro da attacchi di brute force.

Un altro esempio è quella di una code injection in campi non classici, ad esempio tramite l'inserimento password. Per questo caso si potrebbe implementare una sanificazione delle query al sistema, parametrizzando le richieste e impedendo che queste abbiano effetti indesiderati. In più, basandoci sul concetto dei privilegi minimi (1.1) le query potrebbero essere eseguite con i privilegi minimi.

Uno sviluppatore dovrebbe pensare alla sicurezza dall'inizio e durante tutto lo sviluppo

Applicare il concetto della difesa in profondità (1.2), ovvero implementare più livelli di sicurezza

Implementare la sicurezza a livello di applicazione e di infrastruttura

Disabilitare funzionalità o servizi non usati