

→ Top 5 **CWE/SANS**

- ① Improper Neutralization of Special Elements used in an SQL Command (SQL Injection)
- ② Improper Neutralization of Special Elements used in an OS Command (OS Command Injection)
- ③ Buffer Copy without checking Size of Input ('classic Buffer Overflow')
- ④ Improper Neutralization of Input During Web Page Generation (Cross-site Scripting)

⑤ Missing Authentication for Critical Function

① SQL Injection

Il problema alla base, sia in questa vulnerabilità che in molte altre, riguarda il mancato controllo dell'input da parte dell'utente. In questa situazione, l'input dell'utente viene direttamente copiato ed utilizzato in una query SQL al database (comandi come 'DROP table x') o anche più semplicemente modificare il DB.

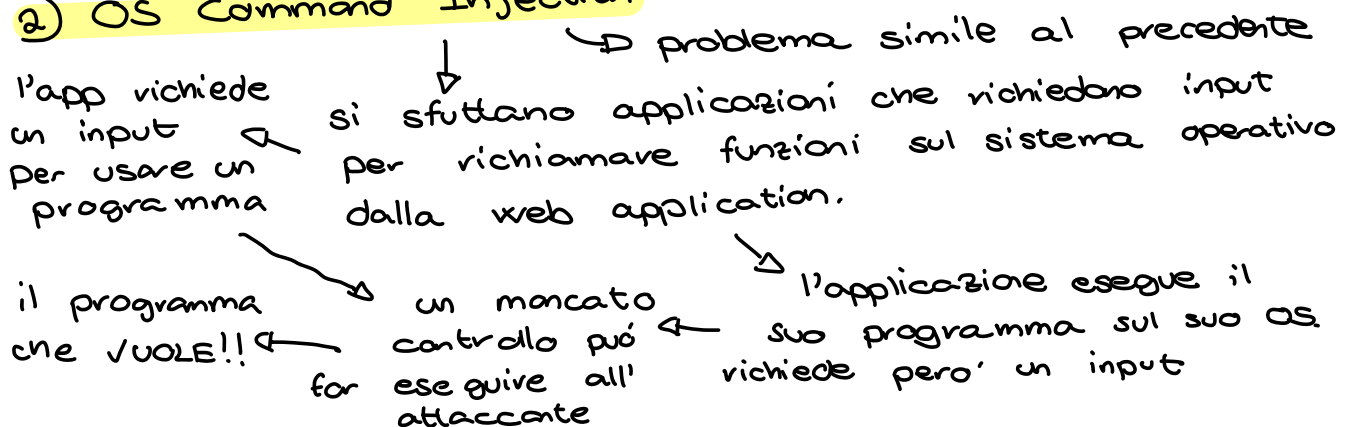
La SQL Injection sfrutta tecnologie di Database.

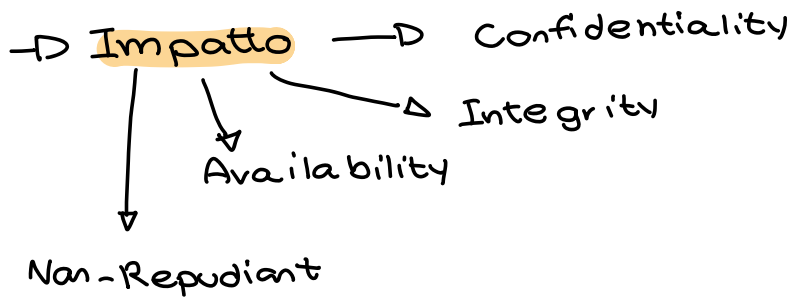
L'impatto e le conseguenze comuni riguardano invece diversi aspetti:

- **Confidentiality**: viene persa la confidenzialità dei dati sensibili. L'attaccante è in grado di leggere i dati dell'applicazione.
- **Access Control**: l'attaccante può accedere ad altri account senza conoscere la password.
- **Integrity**: la modifica o cancellazione dei dati

→ Principale soluzione gestire, controllare e pulire l'input prima di inserirlo in una query.

② OS Command Injection





→ Poiché è l'applicazione di destinazione che esegue direttamente i comandi anziché l'attaccante, è possibile che qualsiasi attività dannosa provenga dall'app o dal proprietario dell'app e venga attribuita ad esso.

3) **Buffer Overflow** → in C, C++

↓
Copro più memoria di quella disponibile

→ Non controllare la dimensione dell'Input prima di copiarlo in un altro buffer in output può portare a vulnerabilità.



→ Inserendo codice malevolo dentro il buffer, è possibile eseguire tale codice dallo scope del programma.

4) **Cross-site scripting** → mancato controllo dell'input

↓ XSS

1. Dati non sicuri vengono inseriti in una applicazione web, generalmente tramite una web request
2. La web app genera contenuto non sicuro
3. Non viene impedito inserimento di JS
4. Visita pagina con script infetto

5. Il web browser della vittima esegue lo script che viene generato dal web server

→ 3 tipologie di XSS → Reflected XSS (or Non-Persistent)
↓
DOM-Based XSS Stored XSS (or Persistent)

→ Le tecnologie vulnerabili → Web Based app

→ Impatto → Confidentiality

↓
Access Control Integrity
↓
Availability

→ L'attacco XSS più comune riguarda la divulgazione dei cookie dell'utente, i quali racchiudono informazioni anche private.

↓
Sfruttare script sul browser per installare virus

↓
Utilizzando i cookie di un altro utente è possibile violare il Controllo degli accessi

5) Missing Authentication for Critical Function

→ Impatto

→ Access control
other

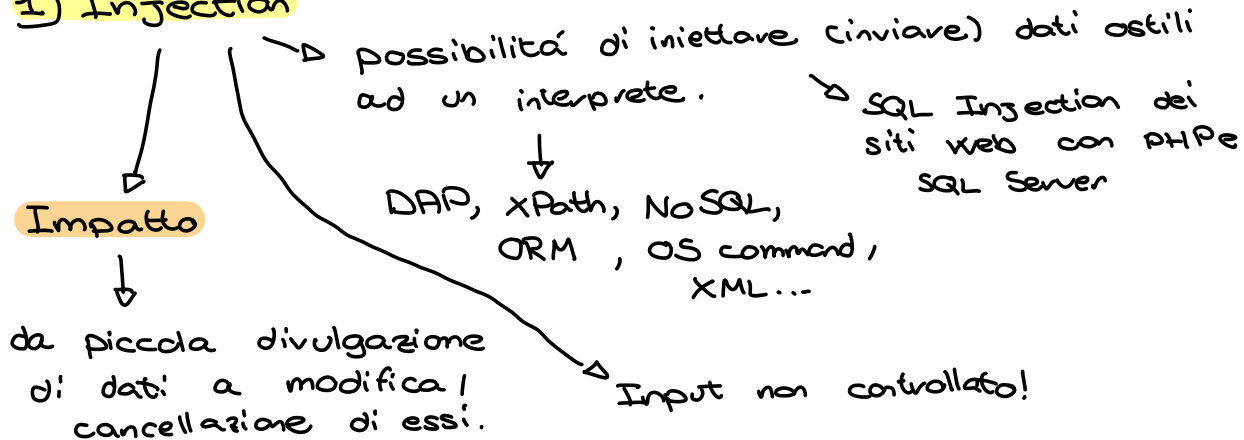
↓
assumere privilegi o assumere l'identità di altri

→ Il software non effettua alcuna autenticazione per le funzionalità che richiedono l'identità verificata

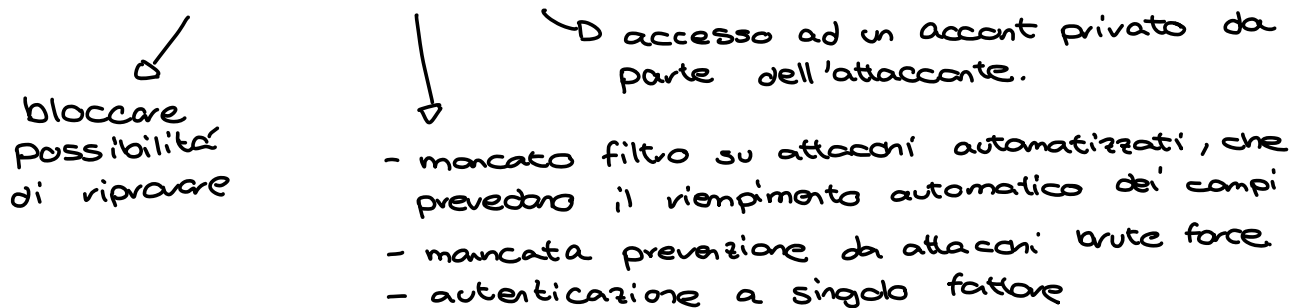
→ Top 5 OWASP

- ① Injection
- ② Broken Authentication
- ③ Cross-site scripting
- ④ Broken Access Control
- ⑤ Security Misconfiguration

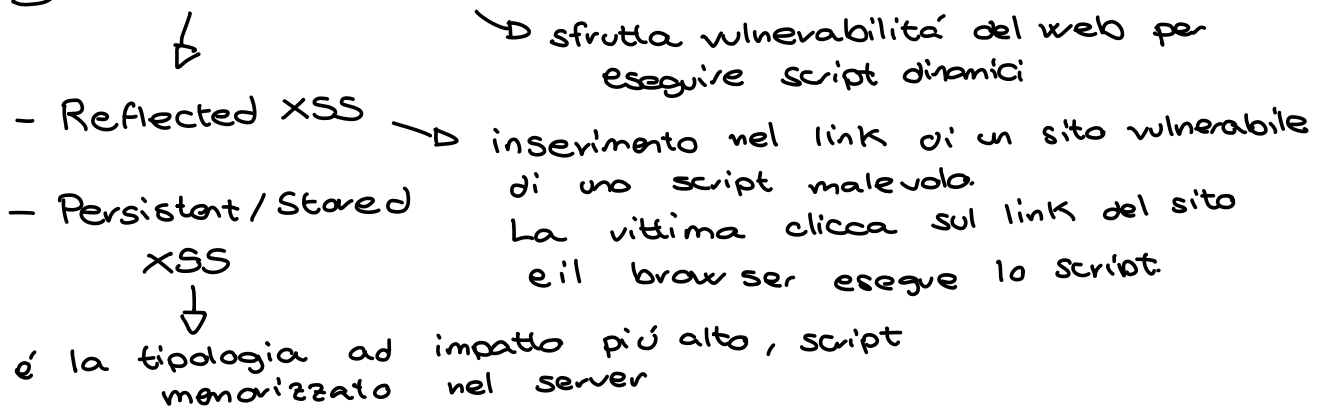
1) Injection



2) Broken Authentication



3) Cross-Site Scripting



- DOM XSS → impatto moderato, il codice viene eseguito direttamente nel framework / JS

4) Broken Access Control

Verifica che alcuni specifici url generalmente nascosti siano accessibili solo da utenti autorizzati

↳ L'attaccante, visto come un utente correttamente autenticato accede ad informazioni al di fuori del suo dominio

5) Security Misconfiguration

credenziali di default

↳ può essere presente ovunque

↳ configurazione errata dei criteri di sicurezza di un'applicazione

↳ permettono all'attaccante di accedere a dati o a funzionalità non autorizzate, tuttavia in casi più rari