

Ora, in questa nostra trattazione, ci concentriamo sulle Web Application (al di sopra della pila ISO/OSI).  
Assumiamo quindi che il WebServer sia sicuro...

OWASP é un ente no-profit che é nata in tale ambito.  
Più nello specifico ha classificato una **Top10** delle vulnerabilità più importanti.

1. **Injection**
2. **XSS (Cross Site Scripting)**
3. **Broken Authentication & Session Management**
4. **Insecure Direct Object References**
5. **CSRF (Cross Site Request Forgery)**
6. **Security Misconfiguration**
7. **Insecure Cryptographic Storage**
8. **Failure to Restrict URL access**
9. **Insufficient Transport Layer Protection**
10. **Unvalidated Redirects and Forwards**

Per classificarle viene usata una metodologia che tiene in considerazione:

- Gli agenti che vogliono sfruttare le vulnerabilità
- I vettori di attacco
- Le vulnerabilità
- Le conseguenze
  - Di tipo tecnico (*tech impact*)
  - In ambito di business (*business impact*)

	Threat agent	Attack vectors	Weakness Prevalence	Weakness Detectability	Tech impact	Business impact
1		easy	common	average	severe	
2		average	very widespread	easy	moderate	
3		average	common	average	severe	
4		easy	common	easy	moderate	
5		average	widespread	easy	moderate	
6		easy	common	easy	moderate	
7		difficult	uncommon	difficult	severe	
8		easy	uncommon	average	moderate	
9		difficult	common	easy	moderate	
10		average	uncommon	easy	moderate	

# 1 - Injection

Per comprendere questo concetto consideriamo un esempio pratico:

- Rete protetta da Firewall e/o Reverse Proxy
- Applicazione per Autenticare Utente (con username e password)
- Un DataBase in cui sono presenti le credenziali di accesso degli utenti e dati sensibili



nome	cognome	codice	carta_di_credito
francesco	bergadano	pipipo	12345678
mario	rossi	pluto	87654321


**Cosa può succedere?** Gli input inviati dall'utente tramite form non sono controllati e quindi, come abbiamo già visto in altri contesti, il testo inserito é del **codice eseguibile**

nome:  cognome:  codice:

**Cosa provoca?**

```
$query = "SELECT * FROM db where  
        nome=' ".$_GET['nome']."' and  
        cognome=' ".$_GET['cognome']."' and  
        codice=' ".$_GET['codice']."'";
```

```
$query = "SELECT * FROM db where  
        nome=' ".$_GET['nome']."' and  
        cognome=' ".$_GET['cognome']."' and  
        codice=' OR nome = 'francesco'";
```



che rende la query true :(

Questo specifico attacco prende il nome di **SQL Injection**

Sicuramente una delle contromisure più ovvie e banali é la così detta **"sanificazione degli input"**

## 2 - XSS (Cross Site Scripting)

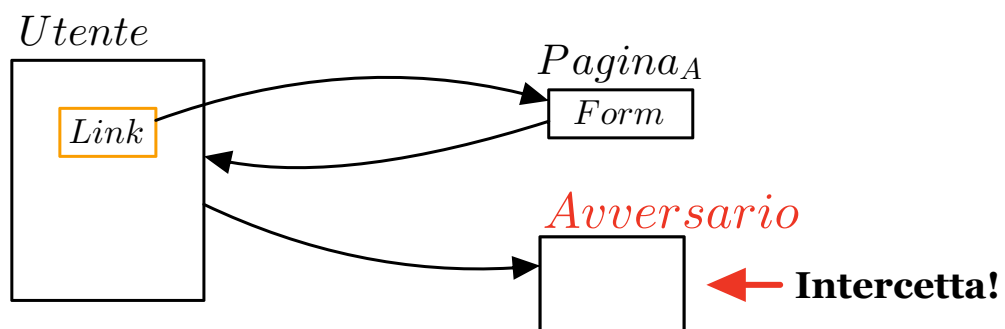
Di seguito parleremo di uno specifico caso di attacco che prende il nome di **XSS - reflector**

Scenario:

- Utente che apre un *link* verso una
- Pagina A (contenente form) ←
- Pagina Avversario

La **vulnerabilità** è proprio in quel link: l'utente (inconsco) cliccando sul link, **inserirà automaticamente uno script** all'interno del codice..

nome:   
cognome:





```
setcookie("TestCookie",1);
echo "INIZIO<br>";
echo "Buongiorno ";
echo $_GET['nome'];
echo "<br><br>";
echo "<br>Valore di TestCookie = ";
echo $_COOKIE["TestCookie"];
echo "<br>FINE<br>";
```

**Attacco:** il browser fa da *reflector* (svolge una specie di echo: Invio "Marco"; lui mi risponde "Buongiorno Marco").

In questo modo però viene eseguito lo script che può ad esempio portare ad intercettare i COOKIE (alla base della gestione delle sessioni) e/o altri parametri del browser. Si pensi ad esempio che i siti li utilizzano per prolungare la sessione in cui si è autenticati

Variante: **XSS - stored**, in cui memorizza i dati da utilizzare poi in un secondo momento...

1. **Injection** -> vedi sopra 
2. **XSS (Cross Site Scripting)** -> vedi sopra 
3. **Broken Authentication & Session Management**  
L'autenticazione dell'utente viene resa vana ed inoltre l'utente non autorizzato riesce ad inserirsi all'interno di una sessione di un utente autorizzato
  - Esempio: non si utilizza funzione di 'log out'
4. **Insecure Direct Object References**  
Un oggetto che è raggiungibile anche fuori dalla sessione
5. **CSRF (Cross Site Request Forgery)**  
Operazioni critiche svolte previa autenticazione ma che non vengono subito terminate dall'utente...
6. **Security Misconfiguration**  
Sistemi mal configurati e/o non aggiornati
7. **Insecure Cryptographic Storage**  
Dati memorizzati in chiaro e/o con cifratura debole. N.B. Se settati male anche le "password salate" (con aggiunta di bit) sono vulnerabili.
8. **Failure to Restrict URL access**  
Con URL modificato É possibile raggiungere oggetti non protetti
9. **Insufficient Transport Layer Protection**  
Pagine protette e altre no; certificati scaduti o non validi...
10. **Unvalidated Redirects and Forwards**  
Indirizzamento a pagine pericolose