- Nelle prime versioni di ASP.NET le stringhe di connessione erano memorizzate nel file web.config. Adesso, le più recenti applicazioni ASP.NET Core possono leggere le configurazioni da varie fonti come appsettings.json, da variabili di ambiente, da argomenti della riga di comando, ecc. È possibile, in pratica, archiviare la stringa di connessione ovunque si voglia. In ogni caso, è sempre meglio che comporla a runtime con l'input dell'utente. Si separa così l'applicazione dai metadati. Il file in questione deve essere messo in sicurezza attivando la modalità "protected configuration", che permette di memorizzare le stringhe di connessione in forma crittografata (encrypted).

Esempio.
Nel seguente codice la stringa di connessione viene letta dal file appsettings.json e la sua composizione non è vulnerabile ad alcuna injection:

```
var builder = new ConfigurationBuilder();
builder.AddJsonFile("appsettings.json", optional: false);
var configuration = builder.Build();
connectionString = configuration.GetConnectionString("SQLConnection");
```

Per ulteriori informazioni: http://cwe.mitre.org/data/definitions/99.html,
CWE-99: Improper Control of Resource Identifiers ('Resource Injection').

7.8.5 LDAP Injection

Come riconoscerla

La LDAP Injection è un tipo di attacco cui sono vulnerabili le applicazioni e che utilizzano l'input, senza verificarlo adeguatamente, per costruire query LDAP (Lightweight Directory Access Protocol).
Se coronato da successo, l'LDAP injection potrebbe consentire un furto di informazioni, un'elevazione dei privilegi e l'autenticazione con un'identità altrui (spoofing).
Per comunicare con la directory delle utenze (ad esempio Active Directory), l'applicazione costruisce dinamicamente delle query. Se utilizza l'input utente senza verificarlo, un malintenzionato può inserire comandi modificati ad arte per carpire informazioni non dovute.

Come difendersi

Validare tutti gli input, indipendentemente dalla loro provenienza. Per la validazione, si consiglia l'approccio white list (sono accettati solo i dati che adottano una struttura specificata nella white list, scartando quelli che non la rispettano). Occorre controllare il tipo del dato, la sua dimensione, l'intervallo di validità (range), il formato ed eventuali valori attesi (white list).
A partire dalla versione 3.5 del Framework .NET, sono state introdotte nella libreria AntiXSS della versione 4 due nuove funzioni che permettono l'encoding delle stringhe da utilizzare per le query LDAP:

- Encoder.LdapFilterEncode. Codifica l'input convertendo i valori non sicuri in \n, dove n rappresenta il carattere non sicuro.
- Encoder.LdapDistinguishedNameEncode. Codifica l'input convertendo i valori non sicuri in #n, dove n rappresenta il carattere non sicuro, mentre i segni ",", "+", "/", "<" e ">" vengono codificati con la barra ("\").

Esempio:
Formato non corretto:

```
ds.Filter = "(&(name=" + input + ")(isPublic=true))"
```

Formato corretto:

```
ds.Filter = "(&(name=" + Encoder.LdapFilterEncode(input) + ")(isPublic=true))"
```

Per ulteriori informazioni: http://cwe.mitre.org/data/definitions/90.html,
CWE-90: Improper Neutralization of Special Elements used in an LDAP Query ('LDAP Injection').