### 7.6.5 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 consisglia 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).

Esempio:
Il seguente codice, avvalendosi di una variabile (nomeutente) pervenuta attraverso l'input, è passbile di LDAP injection, a meno che la stringa non sia sottoposta a codifica (encoding) e validazione.

```
DirectorySearcher search = new DirectorySearcher(de);
search.Filter = "(ACName=" + nomeutente + ")";
search.SearchScope = SearchScope.Subtree;
search.CacheResults = false;
```

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

### 7.6.6 Resource Injection

**Come riconoscerla**

Quando un'applicazione definisce un tipo di risorsa o posizione in base all'input dell'utente, come un nome file o un numero di porta, questi dati possono essere manipolati per eseguire o accedere a risorse diverse.
L'attacco di "path traversal" è un caso particolare della resource injection. In tal caso a essere iniettato è un path manipolativo che punta a risorse diverse nel file system.
Se si utilizza l'input dell'utente per definire la porta sulla quale aprire un socket, si da all'utente la possibilità di introdurre una backdoor attraverso la quale potrebbe prendere il controllo del sistema.

**Come difendersi**

In molti casi non è necessario aprire un socket manualmente; meglio affidarsi a librerie e protocolli esistenti.

- Tutti i dati inviati devono essere crittografati, se sono sensibili. Nel dubbio se i dati siano sensibili o possano diventarlo, meglio comunque crittografarli.
- Qualsiasi input letto dal socket deve essere validato.
- Le applicazioni non dovrebbero utilizzare l'input dell'utente per accedere a risorse del sistema. Nel caso si scelga di farlo, è obbligatorio validare l'input, per esempio attraverso una white list. Se si consente la creazione di socket, controllare scrupolosamente questo tipo di attività.

Esempio:
Creazione di un socket passibile di resource injection, qualora i parametri fossero controllati dall'utente:

```
public static void Run()
{
  Socket socket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
```