Come difendersi

- Rimodulare il codice per evitare una qualsiasi esecuzione diretta di script di comandi. Per effettuare operazioni di sistema, utilizzare eventualmente API fornite dalla piattaforma.
- Se non è possibile fare a meno di lanciare shell dei comandi, assicurarsi tuttavia di eseguire solo stringhe statiche, che non includano l'input dell'utente.
- Validare tutti gli input, indipendentemente dalla loro provenienza. La convalida dovrebbe essere basata su una white list: dovrebbero essere accettati solo i dati che adattano a una struttura specificata, e scartati i dati che non rientrano in questa categoria. I parametri devono essere limitati a un set di caratteri consentito e i caratteri riconosciuti come estranei devono essere filtrati e neutralizzati (escaping). Oltre ai caratteri, occorre controllare il tipo del dato, la sua dimensione, l'intervallo di validità (range), il formato ed eventuali valori attesi (white list).
- Configurare l'applicazione da eseguire utilizzando un account utente limitato che non disponga di privilegi non necessari.
- L'esecuzione del codice dovrebbe utilizzare un account utente separato e dedicato, fornito dei soli privilegi strettamente necessari, in base al principio denominato "Principle of Least Privilege". Il principio stabilisce che agli utenti venga attribuito il più basso livello di "diritti" che possano detenere rimanendo comunque in grado di compiere il proprio lavoro.

Per ulteriori informazioni: http://cwe.mitre.org/data/definitions/77.html,
CWE-77: Improper Neutralization of Special Elements used in a Command ('Command Injection').

7.8.4 Connection String Injection

Come riconoscerla

Questo tipo di attacchi è possibile nel momento in cui l'applicazione affida all'input utente la composiszione dinamica della stringa di connessione al database oppure al server.
Un utente malintenzionato potrebbe manipolare la stringa di connessione dell'applicazione al database oppure al server. Utilizzando strumenti e modifiche di testo semplici, l'aggressore potrebbe essere in grado di eseguire una delle seguenti operazioni:

- Danneggiare le performance delle applicazioni (ad esempio incrementando il valore relativo al MIN POOL SIZE);
- Manomettere la gestione delle connessioni di rete (ad esempio, tramite TRUSTED CONNECTION);
- Dirigere l'applicazione sul database fraudolento anziché a quello genuino;
- Scoprire la password dell'account di sistema nel database (tramite un brute-force attack).

Per comunicare con il proprio database o con un altro server (ad esempio Active Directory), l'applicazione costruisce dinamicamente una sua stringa di connessione. Questa stringa di connessione viene costruita dinamicamente con l'input inserito dall'utente. Se i valori immessi sono stati verificati in misura insufficiente o non sono stati affatto verificati, la stringa di connessione potrebbe essere manipolata ad arte a vantaggio dell'attaccante.

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). In generale, è necessario controllare il tipo del dato, la sua dimensione, l'intervallo di validità (range), il formato ed eventuali valori attesi (white list).
- Evitare di costruire dinamicamente stringhe di connessione. Se è necessario creare dinamicamente una stringa di connessione evitare di includere l'input dell'utente. In ogni caso, utilizzare utilità basate sulla piattaforma, come SqlConnectionStringBuilder di .NET, o almeno codificare l'input validato come il piu' idoneo per la piattaforma utilizzata.