Come difendersi

- Bisogna evitare di incorporare entità esterne.
- Occorre assicurarsi di disabilitare il parser dal caricamento automatico di entità esterne.
- Formati di dati meno complessi, come JSON, possono rendere più difficile la serializzazione di dati sensibili.
- Devono essere apportati i necessari aggiornamenti a tutti i parser e alle librerie XML in uso da parte dell'applicazione o sul sistema operativo sottostante.
- Se viene utilizzato SOAP, occorre aggiornarlo alla versione 1.2 o successive.
- Implementare la convalida dell'input come evidenziato in altri punti.
- Verificare che la funzionalità di caricamento di file XML o XSL convalidI l'XML in entrata utilizzando uno schema XSD.

Esempio**:**

Formato non corretto - Il parsing del documento XML, qui racchiuso nella stringa OurOutputXMLString, carica qualunque entità esterna, se non validata.

```
XmlDocument xmlDoc = new XmlDocument();
xmlDoc.LoadXml(OurOutputXMLString);
```

Formato corretto - La riga evidenziata imposta a null il valore del resolver. In questo modo s'impedirà al parser di prendere in considerazione le entità esterne.

```
XmlDocument xmlDoc = new XmlDocument();
xmlDoc.XmlResolver = null;
xmlDoc.LoadXml(OurOutputXMLString);
```

Per una regolazione più sottile, basterà utilizzare una classe derivata da XmlUrlResolver. Si potrà decidere quali domini potranno essere accettati dfile XML.

7.6.10 Ulteriori indicazioni per lo sviluppo sicuro

Di seguito ulteriori suggerimenti per lo sviluppo sicuro in C#.

7.6.10.1 Managed Wrapper per l'implementazione del codice nativo

Spesso sorge la necessità di rendere disponibile una funzionalità utile in codice nativo per il codice gestito. La realizzazione dei managed wrapper può essere semplificata usando platform invoke o COM interop. Per la riuscita di quest'operazione è tuttavia necessario che i chiamanti dei wrapper dispongano di diritti per il codice non gestito (unmanaged code).
Invece di concedere diritti per il codice non gestito a tutte le applicazioni che usano il wrapper, è preferibile fornire questi diritti solo al codice wrapper. Se la funzionalità sottostante non espone alcuna risorsa e l'implementazione è verosimilmente sicura, chiunque potrà chiamare il wrapper con i regolari diritti sull'unmanaged code. Quando invece, la funzionalità nativa espone delle risorse, il wrapper può mettere i suoi chiamanti in pericolo, per cui è necessaria, da parte sua, un'attenta verifica della sicurezza del codice nativo.

7.6.10.2 Library Code che espone risorse protette

La libreria funge da interfaccia per l'accesso a determinate risorse che non sono altrimenti disponibili; deve quindi richiedere autorizzazioni per l'accesso alle risorse che utilizzano. In generale, laddove si espone una risorsa (qualunque essa sia), il codice deve implementare una richiesta di autorizzazione appropriata alla risorsa (cioè deve eseguire un controllo di protezione).

7.6.10.3 Richieste di autorizzazione

La richiesta di autorizzazioni è il modo in cui si consente al Common Runtime Language (CLR) di .NET di sapere cosa deve fare il codice per eseguire il proprio lavoro. Sebbene la richiesta di autorizzazioni sia facoltativa e non sia necessaria per la compilazione del codice, ci sono importanti motivi per richiedere le