Gli attacchi XPath Injection sono possibili quando un sito Web utilizza le informazioni fornite dall'utente per costruire una query XPath per i dati XML. Inviando informazioni intenzionalmente malformate nel sito Web, un utente malintenzionato può scoprire come sono strutturati i dati XML o accedere a dati a cui normalmente non avrebbe accesso. Potrebbe persino essere in grado di elevare i suoi privilegi sul sito Web se i dati XML vengono utilizzati per l'autenticazione (come un file utente basato su XML).

Come difendersi

- Evitare che la costruzione della query XPath dipenda dalle informazioni inserite dall'utente. Possibilmente mapparla con i parametri utente mantenendo la separazione tra dati e codice. Nel caso fosse necessario includere l'input dell'utente nella query, questo dovrà essere precedentemente validato.
- Validare tutti gli input, indipendentemente dalla loro provenienza. La validazione dovrebbe essere basata su una white list (si dovrebbero accettare solo i dati che adattano a una struttura specificata, 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).

Per ulteriori informazioni si veda: http://cwe.mitre.org/data/definitions/643.html,
CWE-643: Improper Neutralization of Data within XPath Expressions ('XPath Injection').

7.8.9 Ulteriori indicazioni per lo sviluppo sicuro

7.8.9.1 ASP.NET Web Form

Web form è una tecnologia basata su ASP.NET di Microsoft, in cui il codice eseguito sul server genera dinamicamente l'output di pagina Web al browser o al dispositivo client. È uno dei quattro modelli (assieme a ASP.NET MVC, ASP.NET Web Pages, ASP.NET Single Page Applications) di programmazione che possono essere utilizzati per la creazione di applicazioni web ASP.NET.
È compatibile con qualsiasi browser, dispositivo mobile o linguaggio supportato da .NET ed è flessibile in quanto offre la possibilità di aggiungere controlli creati dall'utente e terze parti.

Segue un elenco di best practices per lo sviluppo sicuro.

- **Concatenazione di stringhe**: Utilizzare StringBuilder. Nella concatenazione delle stringhe, l'uso di StringBuilder è preferibile rispetto a String.Concat o all'utilizzo dell'operatore '+'. Più nel dettaglio, StringBuilder è più performante nella concatenazione di un numero elevato di stringhe (>=3), mentre ha prestazioni equiparate a String.Concat per un minor numero (<3) di stringhe.
- **Ajax UpdatePanel**: Evitare chiamate superflue al server. Controllare Page.IsPostBack al caricamento della pagina per assicurarsi che la logica di inizializzazione della pagina venga eseguita quando una pagina viene caricata la prima volta e non in risposta ai postback dei client. Per le convalide, devono essere utilizzati script lato client.
- **ViewState e HiddenFields**: Mantenere i dati minimi in ViewState. ViewState è valido solo per il postback delle stesse pagine: i dati vengono trasmessi al client e restituiti in un campo nascosto. Disattiva ViewState a PageLevel utilizzando EnableViewState.
- **Sessione**: Variabili di sessione
  - Non dovrebbero esistere più di 20 variabili di sessione nel contesto applicativo.
  - Tenere TimeOut di sessione.
  - Disattivare lo stato della sessione, se non si utilizza in una particolare pagina / applicazione.
- **Reindirizzare**: Server.Trasfer vs Response.Redirect. Utilizza Server.Transfer per reindirizzare alle pagine della stessa applicazione e Response.Redirect per reindirizzare verso una pagina esterna o quando è necessario avviare un nuovo contesto.
- **DataReader**: Utilizzare DataReader per il binding dei dati. Se l'applicazione non richiede la memorizzazione nella cache, è possibile utilizzare DataReader.