7.11.1 Client Dom Code Injection

Come riconoscerla

Un attaccante può eseguire codice arbitrario sulla macchina dell'application server. A seconda dei permessi di cui dispone l'applicazione, potrebbe: accedere al database, leggere o modificare dati sensibili; leggere, creare, modificare o cancellare file; aprire una connessione al server dell'attaccante; modificare il contenuto delle pagine; decifrare dati utilizzando le chiavi dell'applicazione; arrestare o avviare i servizi del sistema operativo; organizzare un reindirizzamento verso siti fake (fasulli) per operazioni di phishing; prendere il completo controllo del server.
Accade perché l'applicazione esegue alcune azioni eseguendo codice incluso nei dati in input non opportunamente validati e verificati. In questo caso, il codice non attendibile viene letto dal browser ed eseguito sul lato client.

Come difendersi

Come prima cosa, l'applicazione non dovrebbe eseguire alcun codice non attendibile da qualsiasi fonte esterna possa provenire, inclusi l'input dell'utente, dei file caricati (upload) o un database.
Se è necessario passare dati non attendibili all'esecuzione dinamica, applicare una convalida dei dati molto rigorosa. Come al solito, occorre convalidare tutti gli input, indipendentemente dalla fonte. I parametri devono essere limitati a un set di caratteri consentito e l'input non convalidato deve essere eliminato. Oltre ai caratteri, occorre controllare il tipo di dati, la loro dimensione, l'intervallo di validità, il formato e l'eventuale corrispondenza all'interno dei valori previsti (white list). Sconsigliata invece la black list, ossia un elenco di valori non consentiti: l'elenco sarebbe sempre troppo limitato, rispetto ai casi che potrebbero verificarsi.
Se è assolutamente necessario includere dati esterni nell'esecuzione dinamica, è consentito passare i dati come parametri al codice, ma bisogna evitare assolutamente di eseguire direttamente i dati utente.
L'account con il quale l'applicazione viene avviata deve avere molte restrizioni e non deve godere di privilegi non necessari.
Evitare di creare codice XML o JSON in modo dinamico.
Proprio come la creazione di codice HTML o SQL potrebbero causare dei bug di XML Injection, utilizzare una libreria di codifica o delle librerie JSON o XML affidabili per rendere sicuri gli attributi dei dati degli elementi.
Non eseguire la crittografia nel codice lato client. Utilizzare le tecnologie TLS/SSL e crittografare le informazioni sul server.
Evitare di chiamare dinamicamente una funzione senza averne prima bonificato il codice.

Esempio:

```
var input = document.getElementById("id").value;
window.setInterval( myFunc(input), 1000);
```

Questo il software corretto dopo la sanitizzazione:

```
var input = document.getElementById("id").value;
var trusted = escape(input);
window.setInterval( myFunc(trusted), 1000);
```

Esempio

Uso corretto dell'aggiornamento dinamico dell'HTML nel DOM:

```
document.write("<%=Encoder.encodeForJS(Encoder.encodeForHTML(untrustedData))%>");
```

Nel caso debba essere impostato del codice Javascript per delle chiamate dinamiche, vanno utilizzati solo metodi predefiniti o codice Javascript non influenzabile da variabili dinamiche. Non si deve usare codice con routine tipo "eval()" particolarmente vulnerabili.

Esempio di codice Javascript sicuro:

```
window.setInterval("timedFunction();", 1000);
```