la funzione after() accetta un elemento DOM, quindi consente di creare un nodo di testo:

```
select.after(document.createTextNode(option.text()));
```

Per ulteriori informazioni si veda: http://cwe.mitre.org/data/definitions/97.html,
CWE-97: Improper Neutralization of Server-Side Includes (SSI) Within a Web Page.

7.11.4 Client Dom XSS

Come riconoscerla

Un utente malintenzionato può utilizzare il social engineering per indurre un utente a inviare l'input modificato in modo malevolo verso il sito Web, ad esempio inducendolo a cliccare su un URL con un'ancora (hash) modificata, facendo sì che il browser riscriva le pagine Web. L'aggressore può quindi dirottare la vittima verso un server fake (fasullo), che gli consentirebbe di rubare la password dell'utente, farsi inserire i dati della carta di credito, fornire informazioni false o eseguire del malware. Ovviamente la vittima rimane ignara di ciò che accade.
L'attacco è possibile perché la pagina Web dell'applicazione incorpora nella pagina dati provenienti dall'input dell'utente (incluso l'URL della pagina), facendo sì che il browser li visualizzi come parte della pagina Web. Se l'input include frammenti HTML o JavaScript, anche questi vengono visualizzati (ed eseguiti). La vulnerabilità è il risultato dell'incorporamento di input dell'utente arbitrario senza prima codificarlo in un formato che impedirebbe al browser di trattarlo come HTML anziché come testo normale.

Come difendersi

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.
Effettuare un encoding (codifica) su tutti i dati dinamici prima di includerli nella pagina web. Considerare per tale scopo la libreria ESAPI4JS di OWASP.

Per creare dinamicamente URL in JavaScript, utilizzare la libreria OWASP ESAPI4JS:

```
window.location = ESAPI4JS.encodeForURL(input);
```

Per ulteriori informazioni si veda: http://cwe.mitre.org/data/definitions/97.html,
CWE-97: Improper Neutralization of Server-Side Includes (SSI) Within a Web Page.

7.11.5 Client Resource Injection

Come riconoscerla

Un malintenzionato potrebbe essere in grado di aprire una backdoor che gli consente di connettersi direttamente al server delle applicazioni, portando potenzialmente al controllo del server o ad altri attacchi indiretti. In particolare, modificando il numero di porta del socket, il malintenzionato può essere in grado di bypassare insufficienti controlli di rete o offuscare l'attacco da parte dei dispositivi di rete. Inoltre, questa vulnerabilità può essere sfruttata per bypassare i firewall o altri meccanismi di controllo degli accessi; utilizzare l'applicazione come proxy per la scansione delle porte delle reti interne e l'accesso diretto ai sistemi locali; o indurre erroneamente l'utente a inviare informazioni riservate a un server fasullo.

Come difendersi

Non consentire a un utente, direttamente o indirettamente, di definire i parametri dei socket o altre impostazioni di rete.
Se possibile, limitare i WebSocket agli URL predefiniti.