Node.js: come proteggere ExpressJS dagli attacchi CSRF

Possiamo proteggere ExpressJS contro gli attacchi CSRF usando un modulo NPM specifico.

csurf è un middleware che automaticamente crea e valida un token CSRF che impedisce questo tipo di attacchi sulle richieste HTTP POST.

Possiamo aggiungere il middleware alla nostra app in questo modo.

```
'use strict';

const express = require('express');
const bodyParser = require('body-parser');
const csrf = require('csurf');

const app = express();

app.use(bodyParser.urlencoded({ extended: false }));
app.use(csrf());
```

Le pagine della nostra applicazione devono quindi contenere il token generato affinché le richieste POST possano superare la validazione. Per farlo possiamo sfruttare l'oggetto `response.locals` per la condivisione dei dati con le view.

```
app.use((req, res, next) => {
  res.locals.csrfToken = req.csrfToken();
```

```
  next();
});
```

`req.csrfToken()` è il metodo aggiunto dal middleware all'oggetto `request` e serve appunto a generare e reperire il token per la richiesta corrente.

A questo punto possiamo inserire il token come campo nascosto nei form avente come attributo `name` il valore `_csrf`.

```
<form action="/contact" method="post">
  <input type="hidden" name="_csrf" value="<%=
csrfToken %>">
  <!-- ... -->
</form>
```

Se le richieste vengono effettuate tramite AJAX si può adottare lo stesso approccio usato in Laravel, ossia impostare per prima cosa un meta tag specifico nelle pagine.

```
<meta name="csrf-token" content="<%= csrfToken %>">
```

Quindi ciascuna richiesta POST dovrà avere l'header HTTP `CSRF-Token` impostato sul valore corrente del token.

```
'use strict';

const postRequest = (url, body) => {
    const token =
document.querySelector('meta[name="csrf-
token"]').getAttribute('content');

    return fetch(url, {
```

```
        credentials: 'same-origin',
        headers: {
                'CSRF-Token': token
            },
              method: 'POST',
              body: body
    });
};
```

Come gestiamo l'errore risultante da un token non valido? Questo middleware solleva un errore che ha la proprietà `code` impostata sul valore `EBADCSRFTOKEN`. Possiamo quindi usare il middleware per la gestione degli errori di ExpressJS.

```
app.use((error, req, res, next) => {
    if(err.code === 'EBADCSRFTOKEN') {
        return res.sendStatus(403);
    }
    return next(error);
});
```

Come si può notare, questo middleware si rivela estremamente efficace nel suo compito e relativamente semplice da usare.