GABRIELE ROMANATO

Implementare la protezione CSRF in Node.js da zero

Il Cross-Site Request Forgery (CSRF) è un attacco che induce un utente autenticato a inviare una richiesta indesiderata a un'applicazione web. Per proteggere una applicazione Node.js da questo tipo di attacco, possiamo implementare da zero un sistema di token CSRF. Questo articolo mostra passo dopo passo come farlo.

1. Generare un Token CSRF

Un token CSRF deve essere:

- Unico per ogni sessione
- Difficile da indovinare

Usiamo il modulo `crypto` di Node.js:

```
const crypto = require('crypto');

function generateCSRFToken() {
  return crypto.randomBytes(32).toString('hex');
}
```

2. Memorizzare il Token nella Sessione

Utilizzeremo `express-session` per gestire la sessione dell'utente:

```
const session = require('express-session');
const express = require('express');
const app = express();

app.use(session({
  secret: 'una_stringa_segretissima',
```

```
  resave: false,
  saveUninitialized: true
}));
```

Ogni volta che un utente carica un form, assegniamo un nuovo token se non esiste già:

```
app.use((req, res, next) => {
  if (!req.session.csrfToken) {
    req.session.csrfToken = generateCSRFToken();
  }
  next();
});
```

3. Includere il Token nel Form HTML

Il token viene inserito in un campo nascosto:

```
<form method="POST" action="/submit">
  <input type="hidden" name="csrfToken" value="<%= csrfToken
%>">
  <input type="text" name="message">
  <button type="submit">Invia</button>
</form>
```

Nel controller della rotta che restituisce il form, passiamo il token:

```
app.get('/form', (req, res) => {
  res.render('form', { csrfToken: req.session.csrfToken });
});
```

4. Validare il Token sul Server

Quando il form viene inviato, il server deve confrontare il token ricevuto con quello memorizzato in sessione:

```
app.post('/submit', express.urlencoded({ extended: false }),
(req, res) => {
  const tokenFromForm = req.body.csrfToken;
  if (tokenFromForm !== req.session.csrfToken) {
    return res.status(403).send('Token CSRF non valido');
  }
  // Proseguire con la logica della richiesta
  res.send('Richiesta valida');
});
```

5. Rigenerare il Token Dopo Ogni Richiesta

Per una maggiore sicurezza, possiamo rigenerare il token CSRF dopo ogni richiesta valida:

```
req.session.csrfToken = generateCSRFToken();
```

Conclusione

Abbiamo implementato un sistema CSRF semplice e funzionante da zero in Node.js, senza dipendere da librerie esterne dedicate. Questo approccio garantisce un buon livello di protezione per form autenticati in ambienti dove si desidera controllo completo sull'implementazione.