

# Progetto Finale CyberSecurity CyFiPr

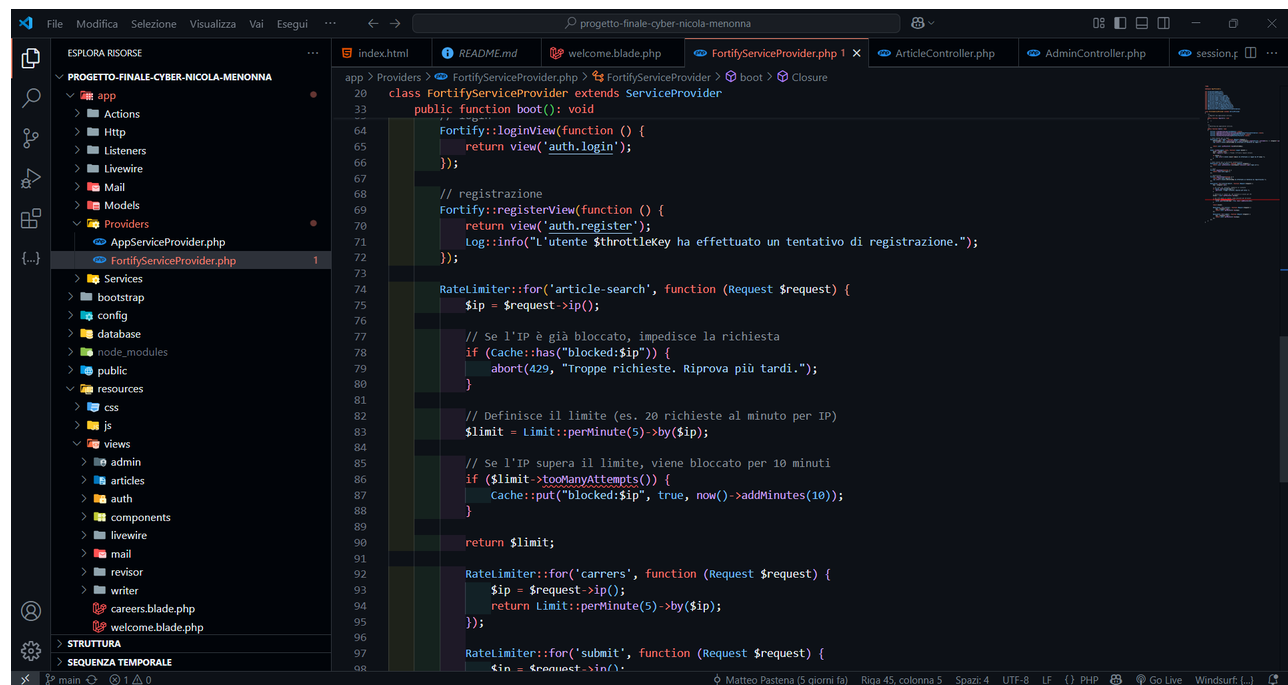
## CHALLENGE 1

Problema:

La challenge 1 comprendeva di aggiungere un sistema di sicurezza che evitasse un qualunque Dos Attack e quindi un'attacco che grazie ad un altissimo numero di richieste facesse crashare il Server.

Mitigato:

Ho aggiunto dei Rate Limiter nel FortifyService che comprendesse le richieste di registrazione login e altri tipi, così facendo se si supera un massimo di richieste (impostato a 5 o 10) giungerà il blocco.



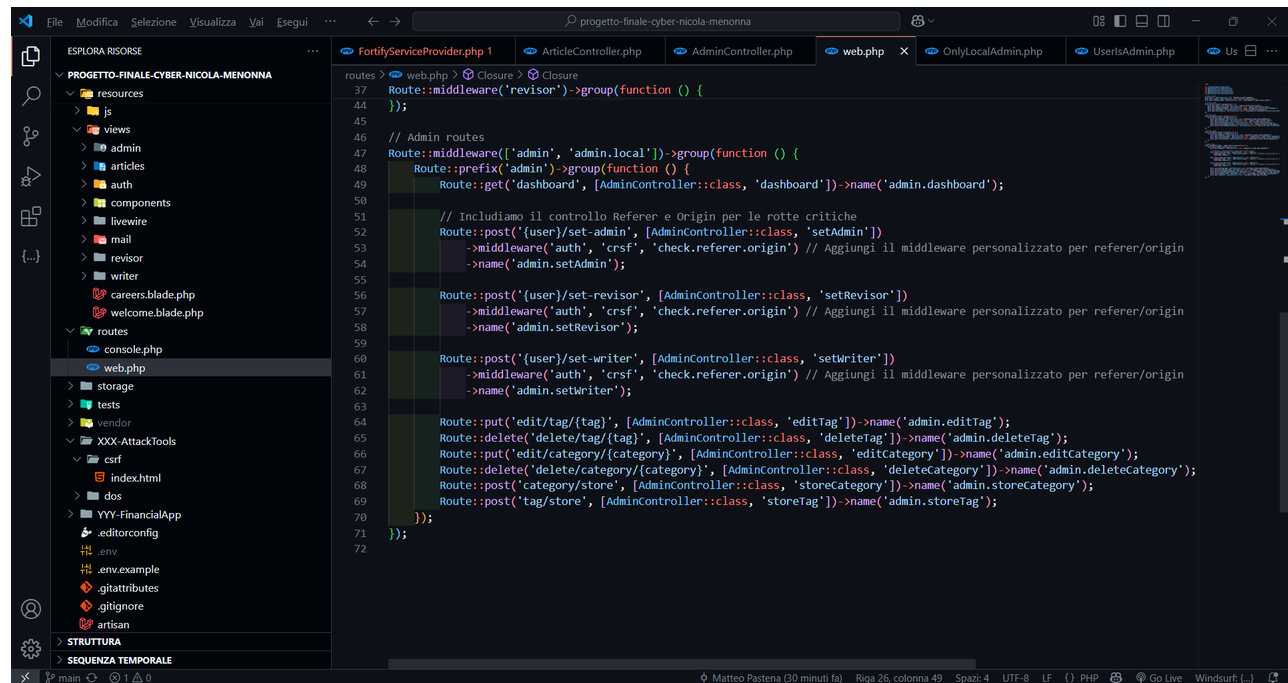
## CHALLENGE 2

Problema:

All'interno del sito alcune funzionalità erano state inserite con le rotte GET che sono deboli dato che inviano semplicemente la richiesta superando il controllo CSRF, E sono facili da sfruttare per un Hacker che può scegliere di attaccare con Ajax Request.

Mitigato:

Sono state cambiate le rotte da GET a POST e sono state incluse in Middleware Group. E' stato inserito un controllo CSRF



```
routes > web.php > Closure > Closure
37 Route::middleware('revisor')->group(function () {
44 });
45
46 // Admin routes
47 Route::middleware(['admin', 'admin.local'])->group(function () {
48 Route::prefix('admin')->group(function () {
49 Route::get('dashboard', [AdminController::class, 'dashboard'])->name('admin.dashboard');
50
51 // Includiamo il controllo Referer e Origin per le rotte critiche
52 Route::post('{user}/set-admin', [AdminController::class, 'setAdmin'])
53     ->middleware('auth', 'csrf', 'check.referer.origin') // Aggiungi il middleware personalizzato per referer/origin
54     ->name('admin.setAdmin');
55
56 Route::post('{user}/set-revisor', [AdminController::class, 'setRevisor'])
57     ->middleware('auth', 'csrf', 'check.referer.origin') // Aggiungi il middleware personalizzato per referer/origin
58     ->name('admin.setRevisor');
59
60 Route::post('{user}/set-writer', [AdminController::class, 'setWriter'])
61     ->middleware('auth', 'csrf', 'check.referer.origin') // Aggiungi il middleware personalizzato per referer/origin
62     ->name('admin.setWriter');
63
64 Route::put('edit/tag/{tag}', [AdminController::class, 'editTag'])->name('admin.editTag');
65 Route::delete('delete/tag/{tag}', [AdminController::class, 'deleteTag'])->name('admin.deleteTag');
66 Route::put('edit/category/{category}', [AdminController::class, 'editCategory'])->name('admin.editCategory');
67 Route::delete('delete/category/{category}', [AdminController::class, 'deleteCategory'])->name('admin.deleteCategory');
68 Route::post('category/store', [AdminController::class, 'storeCategory'])->name('admin.storeCategory');
69 Route::post('tag/store', [AdminController::class, 'storeTag'])->name('admin.storeTag');
70
71 });
72 });
```

## CHALLENGE 3

Problema:

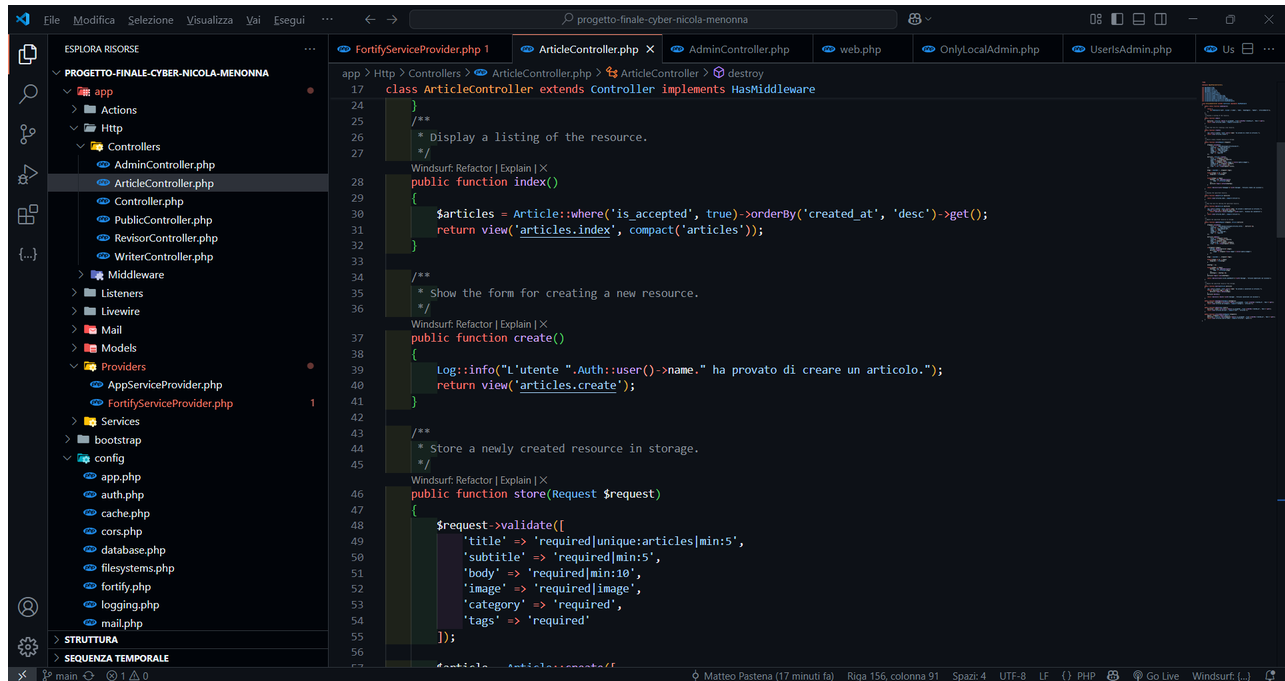
Per una maggiore sicurezza sono stati richiesti dei log per alcune funzionalità che tenessero traccia degli utenti che eseguono determinate azioni:

- login/registrazione/logout
- creazione/modifica/eliminazione articolo
- assegnazione/cambi di ruolo

Mitigato:

sono stati aggiunti dei LOG alle seguenti richieste in FortifyService, AdminController, ArticleController.

come si può vedere dalla foto a riga 39



## CHALLENGE 4

Problema:

All'interno del sito c'è una funzionalità per prendere ispirazione durante la scrittura di un'articolo. Tuttavia è molto facilmente alterabile tramite anche solo ispeziona Elemento e eseguire quindi un attacco SSRF.

Mitigato:

Sono state cambiate le key delle News. sono state mitigate con una Function FetchNews in LatestNews.php, poi sono state sostituite nella sua corrispondenza html Blade in Latest-News.Blade.php togliendo gli url e inserendo solo le chiavi. Infine è stato inserito un controllo di sicurezza nel GetRequest del file HttpService dopo il ParsedUrl.

File Modifica Selezione Visualizza Vai Esegui ... progetto-finale-cyber-nicola-menonna

ESPLORA RISORSE

- PROGETTO-FINALE-CYBER-NICOLA-MENONNA
  - config
    - scout.php
    - services.php
    - session.php
  - database
  - node\_modules
  - public
  - resources
    - css
    - js
    - views
  - admin
  - articles
  - auth
  - components
  - livewire
    - latest-news.blade.php
  - mail
  - revisor
  - writer
  - careers.blade.php
  - welcome.blade.php
- routes
  - console.php
  - web.php
- storage
- tests
- vendor
- XXX-AttackTools
- csrf

- STRUTTURA
- SEQUENZA TEMPORALE

resources > views > livewire > latest-news.blade.php > div > form > div.d-flex > select#apiSelect-form-select > option

```
1 <div>
2 <h3>Articles suggestions for you, get inspired!</h3>
3 <form wire:submit="fetchNews">
4 <label for="apiSelect">Breaking news around the world</label>
5 <div class="d-flex">
6 <select wire:model="selectedApi" id="apiSelect" class="form-select">
7 <option value="">Choose country</option>
8 <option value="NewsAPI-IT">NewsAPI - IT</option>
9 <option value="NewsAPI-UK">NewsAPI - UK</option>
10 <option value="NewsAPI-US">NewsAPI - US</option>
11 </select>
12 <button type="submit" class="btn btn-info">Go</button>
13 </div>
14 </form>
15 <div>
16 @if(isset($news['error']))
17 <p>{{ $news['error'] }}</p>
18 @elseif(isset($news['articles']))
19 @foreach($news['articles'] as $article)
20 <div class="news-article">
21 <h4>{{ $article['title'] }}</h4>
22 <p>{{ $article['description'] }}</p>
23 <a href="{{ $article['url'] }}" target="_blank">Read more</a>
24 </div>
25 @empty
26 <h3>No articles around you</h3>
27 @endforelse
28 @endif
29 </div>
30 </div>
31
```

Riga 10, colonna 42 Spazi: 4 UTF-8 LF {} Blade Go Live Windsurf: (L)

File Modifica Selezione Visualizza Vai Esegui ... progetto-finale-cyber-nicola-menonna

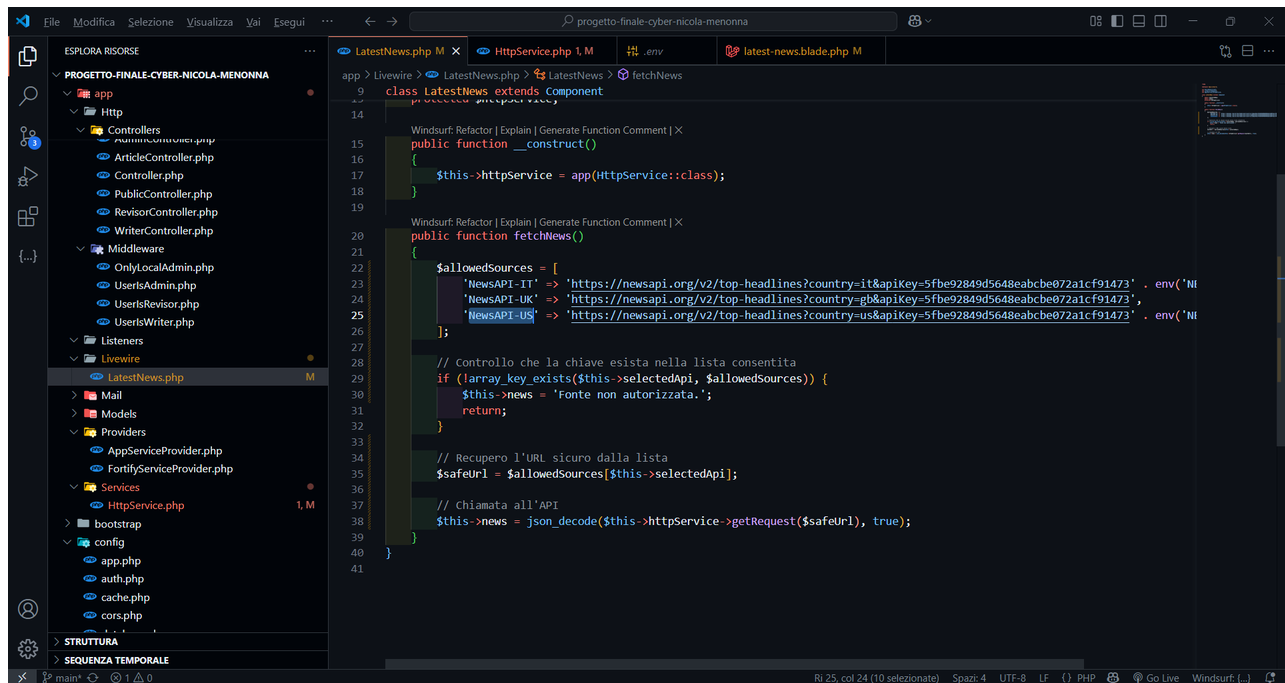
ESPLORA RISORSE

- PROGETTO-FINALE-CYBER-NICOLA-MENONNA
  - app
    - Http
    - Middleware
      - UsersAdmin.php
      - UsersRevisor.php
      - UsersWriter.php
    - Listeners
    - Livewire
      - LatestNews.php
    - Mail
    - Models
    - Providers
      - AppServiceProvider.php
      - FortifyServiceProvider.php
    - Services
      - HttpService.php
    - bootstrap
    - config
      - app.php
      - auth.php
      - cache.php
      - cors.php
      - database.php
      - filesystems.php
      - fortify.php
      - logging.php
      - mail.php
      - queue.php
      - scout.php
  - STRUTTURA
  - SEQUENZA TEMPORALE

app > Services > HttpService.php > HttpService > getRequest

```
9 class HttpService
10 protected $refererHeader; // intestazione Referer
11
12 Windsurf: Refactor | Explain | Generate Function Comment | X
13 public function __construct()
14 {
15     $this->refererHeader = config('app.url');
16     $this->client = new Client();
17 }
18
19 Windsurf: Refactor | Explain | Generate Function Comment | X
20 public function getRequest($url)
21 {
22     $parsedUrl = parse_url($url);
23
24     if (Str::startsWith($url, 'http://internal.finance') && auth()->user()->role !== 'admin') {
25         abort(403, 'Non sei autorizzato ad accedere a questa risorsa interna.');
```

Riga 26, colonna 14 Spazi: 4 UTF-8 LF {} PHP Go Live Windsurf: (L)



## CHALLENGE 5

Problema:

Tramite un Tool BurpSuite è possibile manomettere una richiesta di tipo anche POST per modificarla inserendo dentro di esso uno script malevolo per un'attacco informatico

⚡

Burp

Project

Intruder

Repeater

View

Help

Burp Suite Community Edition v2025.2.4 - Temporary Pro...

—

□

×

Dashboard

Target

Proxy

Intruder

Repeater

Collaborator

Sequencer

Decoder

Settings

Comparer

Logger

Organizer

Extensions

Learn

Intercept

HTTP history

WebSockets history

Match and replace

Proxy settings

Intercept on

→ Forward all

Drop

Open browser

10:56:2... HTTP → Request POST http://cyber.blog:8000/writer/articles/store

10:56:3... WS → To server 15 http://[::1]:5173/

Request

PrettyRawHex

1 POST /writer/articles/store HTTP/1.1

2 Host: cyber.blog:8000

3 Content-Length: 2101189

4 Cache-Control: max-age=0

5 Accept-Language: it-IT,it;q=0.9

6 Origin: http://cyber.blog:8000

7 Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryIuvzZy6NEHm87MH

8 Upgrade-Insecure-Requests: 1

9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36

10 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.0

Inspector

Back

subtitle

Value

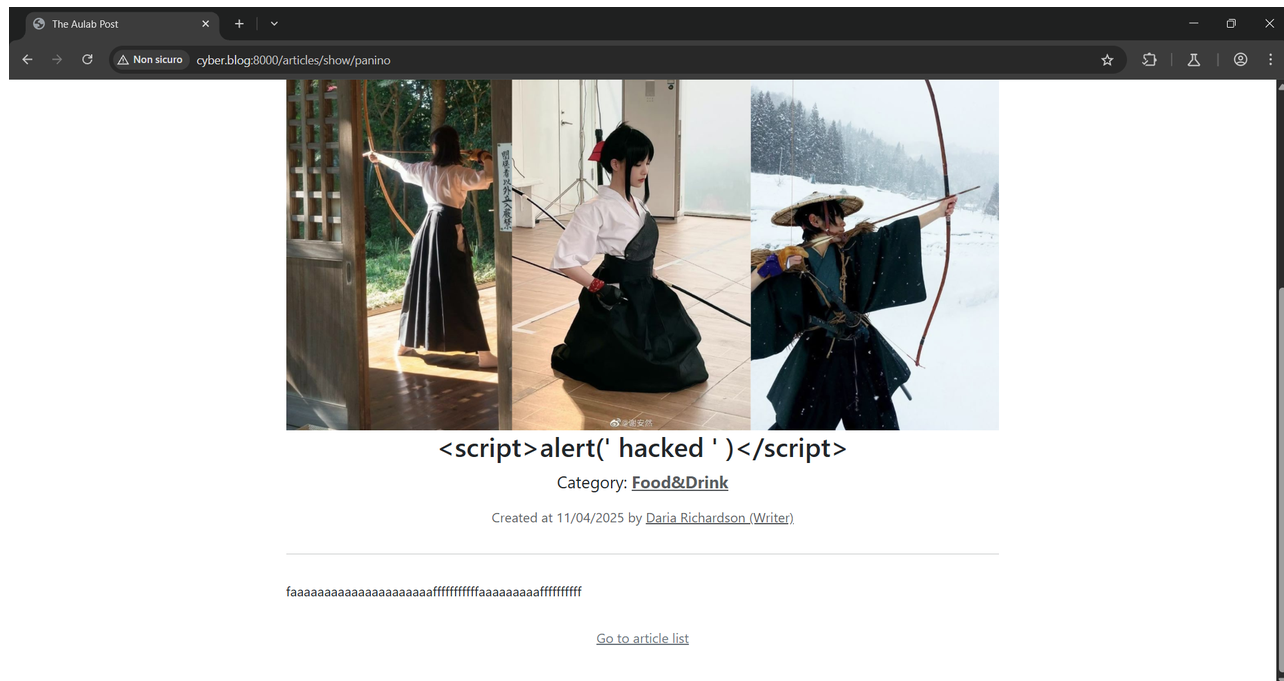
<script>alert(' \r \n hacked \r \n ' \r \n )</script>

Decoded from: URL encoding

<script>alert(' \r \n

Event log (2) All issues

Memory: 391.1MB Disabled



Mitigato:

Ho creato un meccanismo che filtri il testo prima di salvarlo e per essere sicuri anche in fase di visualizzazione dell'articolo. in ArticleController ho aggiunto una public function per sanificare che consente solo determinati tag, e rimuove tutto il resto, compresi `<script>` e attributi tipo `onerror`.

## CHALLENGE 6

Il problema:

Dovuto a scarsa padronanza del framework, non si è ben compreso come funziona il meccanismo offerto dai modelli che gestisce i campi che devono ricevere i dati direttamente dagli utenti tipicamente attraverso i form. Attacco Un utente malevolo può provare a indovinare campi tipici di ruoli utente tipo `isAdmin`, `is_admin` etc.. alterando il form dal browser (Mass Assignment Attack) in questo caso producendo una privilege escalation.

Da mitigare:

Ho specificato in `user.php` nel middleware i campi da modificare inserendo una whitelist che impedisca l'assegnazione automatica di campi non autorizzati come `is_admin`.

in AppService Provider ho aggiunto un'impostazione che trasforma gli errori silenziosi in eccezioni.

Ho creato un ControllerUser a livello generico e al suo interno ho inserito

- **Validazione dei dati**

- **Recupera l'utente autenticato**
- **Filtra i dati**
- **Gestione della password**
- **Aggiorna i dati**
- **Reindirizzamento**

Ho definito la rotta dell'UserController in web.php nella sezione Public Routes.

Ho aggiornato il form HTML affinché l'aggiornamento del profilo punti alla rotta corretta e utilizzi il metodo PUT.