

## Progetto Finale Cyber Security

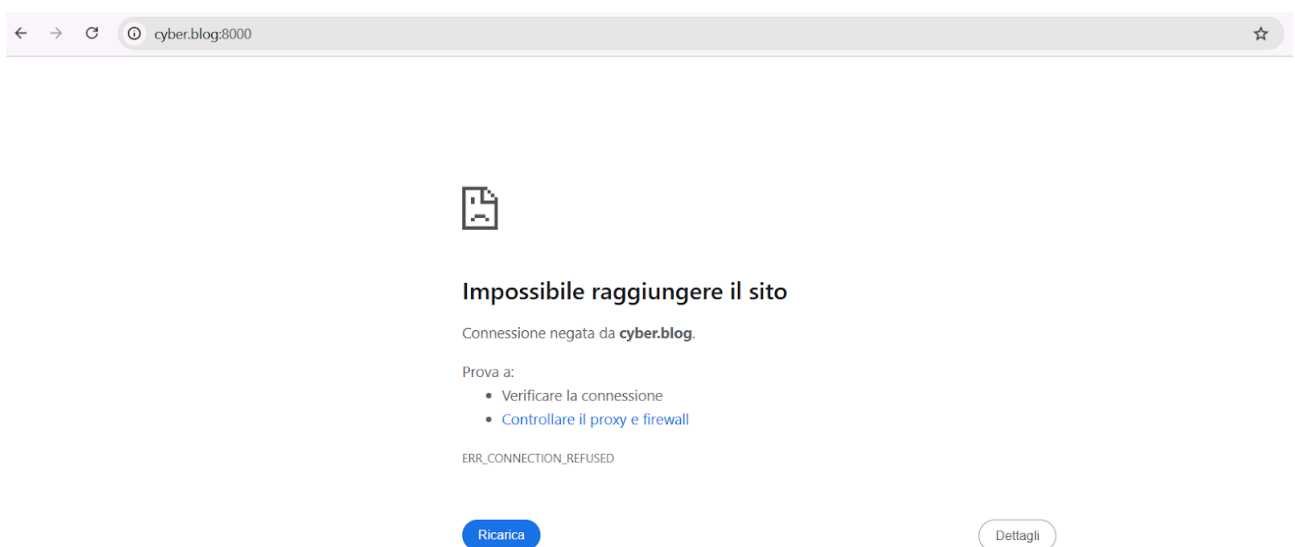
### Challenge 1: Rate limiter mancante

#### Scenario:

Creare ed eseguire uno script (es. in bash con curl) che lancia moltissime richieste sulla stessa rotta con il pericolo di un denial of service

➔ **Prova iniziale dell'attacco Dos sulla rotta "/articles/search":**

```
teres@LAPTOP-9EJ9N6SH MINGW64 ~/wa/specializzazione/progettoFinaleCS/final-project-cyber-blog/XXX-AttackTools/dos (main)
$ ./challenge1.sh
Inizio attacco DoS simulato a http://cyber.blog:8000/articles/search con 10000 richieste...
Attacco DoS simulato completato.
```



#### Mitigazione:

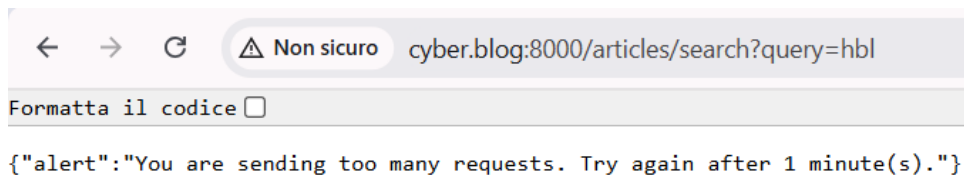
- **Verificare se è già presente un meccanismo di rate limiting per il sistema di autenticazione usato (es. fortify) e nel caso adattarlo alle proprie esigenze**
- **Rate limiter su /careers/submit**
- **Rate limiter su /article/search**
- **Rate limiter globale**
- **Includere il blocco temporaneo per Ip**

➔ **Svolgimento mitigazione:**

1. in **FortifyServiceProvider.php** è già presente un Rate limiter di default per la login e la two-factory; permette 5 tentativi al minuto per ciascuna combinazione di IP e username.
2. creazione di un **middleware** (BlockSuspiciousIPs) da implementare in web.php

3. L'attacco DoS funziona su una qualsiasi rotta pubblica cambiando l'url nel file challenge1.sh; quindi occorre proteggere tutte le rotte pubbliche.
4. creazione di un **middleware-group** in cui inserire le rotte pubbliche da proteggere
5. **blocco temporaneo degli indirizzi IP**: sempre all'interno del middleware BlockSuspiciousIPs è stata inserita una condizione tale per cui se da uno stesso indirizzo IP vengono effettuate più di 30 richieste in 1 minuto, allora quell'indirizzo IP verrà bloccato per 1 minuto.

→ **Prova dell'attacco DoS dopo la mitigazione:**



Non sicuro cyber.blog:8000/articles/search?query=hbl

Formatta il codice ☐

```
{"alert": "You are sending too many requests. Try again after 1 minute(s)."}"
```

```
teres@LAPTOP-9EJ9N6SH MINGW64 ~/wa/specializzazione/progettoFinaleCS/final-project-cyber-blog/XXX-AttackTools/dos
main)
$ ./challenge1.sh
Inizio attacco DoS simulato a http://cyber.blog:8000/articles/search con 5000 richieste...
❌ Attacco bloccato alla richiesta numero 31 con errore 429 (Too Many Requests)
```

```
teres@LAPTOP-9EJ9N6SH MINGW64 ~/wa/specializzazione/progettoFinaleCS/final-project-cyber-blog/XXX-AttackTools/dos
main)
$ ./challenge1.sh
Inizio attacco DoS simulato a http://cyber.blog:8000/articles/search con 5000 richieste...
❌ Attacco bloccato alla richiesta numero 31 con errore 429 (Too Many Requests)
```

```
teres@LAPTOP-9EJ9N6SH MINGW64 ~/wa/specializzazione/progettoFinaleCS/final-project-cyber-blog/XXX-AttackTools/dos
main)
$ ./challenge1.sh
Inizio attacco DoS simulato a http://cyber.blog:8000/articles/index con 5000 richieste...
❌ Attacco bloccato alla richiesta numero 31 con errore 429 (Too Many Requests)
```

## Challenge 2: Operazioni critiche in get

### Scenario:

Ci si espone a possibili attacchi CSRF portando in questo caso ad una vertical escalation of privileges.

Provare un attacco csrf creando un piccolo server php che visualizzi una pagina html in cui in background scatta una chiamata ajax ad una rotta potenzialmente critica e non protetta (es. /admin/{user}/set-admin).

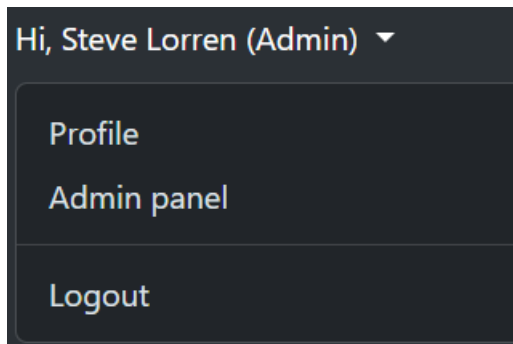
Partendo dal browser dell'utente è possibile che l'azione vada in porto in quanto l'utente ha i privilegi adeguati.

### → Prova iniziale di un attacco csrf:

Utente che deve diventare admin inserito nel db:

7	Kevin Ross (Attacker)	kvrs@gmail.com	0	0	0	NULL
---	-----------------------	----------------	---	---	---	------

Inviare l'html malevolo (href=<http://internal.admin:8000/admin/7/set-admin>) ad un utente admin loggato, quindi in una sessione attiva nella piattaforma target:



Attacco andato a buon fine:

7	Kevin Ross (Attacker)	kvrs@gmail.com	1	0	0	NULL
---	-----------------------	----------------	---	---	---	------



### Mitigazione:

- Cambiare le rotte incriminate da get a patch, eliminando il link nella vista rimpiazzandoli con un form.

### → Svolgimento mitigazione:

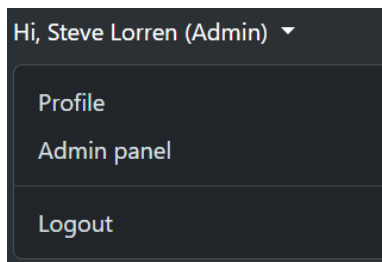
1. Modificate le rotte:
  - /admin/{user}/set-admin
  - /admin/{user}/set-revisor
  - /admin/{user}/set-writerda GET a PATCH
2. Nei components presenti in admin/dashboard.blade.php:  
eliminato i link delle rotte menzionate prima e inserito un form con method= "POST"  
e csrf token.

### → Prova dell'attacco csrf dopo la mitigazione:

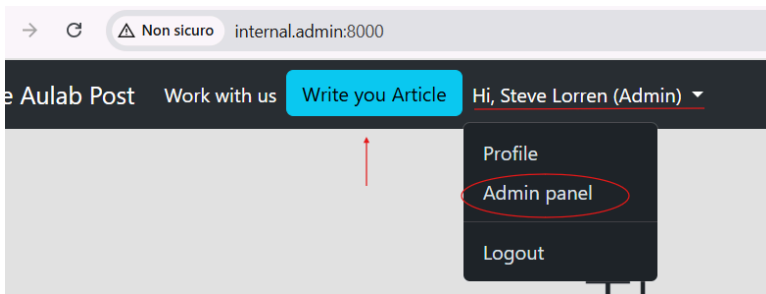
Attaccante che vuole diventare admin:

7	Kevin Ross (Attacker)	kvrs@gmail.com	0	0	0	NULL
---	-----------------------	----------------	---	---	---	------

Utente admin in sessione:



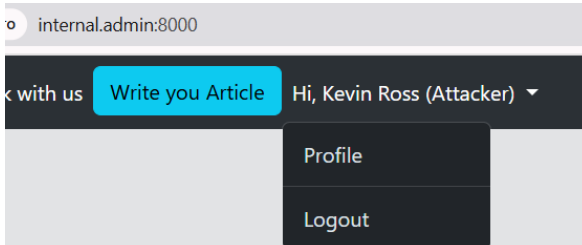
Attacco non andato a buon fine, infatti la navbar dell'admin:



l'utente nel db dopo l'attacco:

7	Kevin Ross (Attacker)	kvrs@gmail.com	0	0	0	NULI
---	-----------------------	----------------	---	---	---	------

La navbar dell'attaccante:



### Challenge 3: Logs mancanti per operazioni critiche

#### Scenario:

Sui tentativi precedenti di DoS non si può risalire al colpevole violando il principio di accountability e no repudiation.

#### Mitigazione:

Log di:

- login/registrazione/logout → inserita parte relativa a Log::info in FortifyServiceProvider.php
- creazione/modifica/eliminazione articolo → inserita parte relativa a Log::info in ArticleController.php
- assegnazione/cambi di ruolo → inserita parte relativa a Log::info in AdminController.php

- + inserita parte relativa a Log::info in RevisorController.php per quanto riguarda la creazione, modifica ed eliminazione degli articoli.

➔ **Scenario dopo la mitigazione in storage/logs/laravel.log :**

```
5534 [2025-02-13 20:13:25] local.INFO: Utente loggato con successo {"username":"user@aulab.it","ip":"127.0.0.1","azione":"login"}
5535 [2025-02-13 20:16:51] local.INFO: Utente disconnesso {"username":"user@aulab.it","ip":"127.0.0.1","azione":"logout"}
5536 [2025-02-13 20:17:16] local.INFO: Nuova registrazione utente {"username":"teresa@email.it","ip":"127.0.0.1","azione":"registrazione"}
5537 [2025-02-13 20:17:16] local.INFO: Utente loggato con successo {"username":"teresa@email.it","ip":"127.0.0.1","azione":"login"}
5538 [2025-02-13 20:17:36] local.INFO: Utente disconnesso {"username":"teresa@email.it","ip":"127.0.0.1","azione":"logout"}
5539 [2025-02-13 20:17:55] local.INFO: Utente loggato con successo {"username":"super.admin@aulab.it","ip":"127.0.0.1","azione":"login"}
5540 [2025-02-13 20:18:33] local.INFO: Utente loggato con successo {"username":"super.admin@aulab.it","ip":"127.0.0.1","azione":"login"}
5541 [2025-02-13 20:24:11] local.INFO: Articolo riportato in revisione {"article_id":1,"title":"Titolo dell'articolo 1","azione":"in revisione","revisore":1}
5542 [2025-02-13 20:24:14] local.INFO: Articolo riportato in revisione {"article_id":2,"title":"titolo articolo 2","azione":"in revisione","revisore":1}
5543 [2025-02-13 20:24:30] local.INFO: Articolo accettato {"article_id":2,"title":"titolo articolo 2","revisore":"Mario Bianchi (Super admin)"}
5544 [2025-02-13 20:24:35] local.INFO: Articolo accettato {"article_id":1,"title":"Titolo dell'articolo 1","revisore":"Mario Bianchi (Super admin)"}
5545 [2025-02-13 20:25:03] local.INFO: Articolo riportato in revisione {"article_id":2,"title":"titolo articolo 2","azione":"in revisione","revisore":1}
5546 [2025-02-13 20:25:09] local.INFO: Articolo rifiutato {"article_id":2,"title":"titolo articolo 2","azione":"rifiutato","revisore":"Mario Bianchi (Super admin)"}
5547 [2025-02-13 20:25:14] local.INFO: Articolo riportato in revisione {"article_id":1,"title":"Titolo dell'articolo 1","azione":"in revisione","revisore":1}
5548 [2025-02-13 20:25:38] local.INFO: Utente disconnesso {"username":"super.admin@aulab.it","ip":"127.0.0.1","azione":"logout"}
5549 [2025-02-13 20:25:57] local.INFO: Utente loggato con successo {"username":"writer@aulab.it","ip":"127.0.0.1","azione":"login"}
5550 [2025-02-13 20:27:39] local.INFO: Articolo creato {"article_id":3,"title":"titolo articolo 3","user_id":3,"ip":"127.0.0.1","timestamp":"2025-02-13 20:27:39"}
5551 [2025-02-13 20:28:50] local.INFO: Articolo modificato {"article_id":2,"title":"titolo articolo 2","user_id":3,"ip":"127.0.0.1","timestamp":"2025-02-13 20:28:50"}
5552 [2025-02-13 20:29:05] local.INFO: Articolo eliminato {"article_id":2,"title":"titolo articolo 2","user_id":3,"ip":"127.0.0.1","timestamp":"2025-02-13 20:29:05"}
5553
```

```
[2025-02-14 14:48:17] local.INFO: Utente loggato con successo {"username":"teresa@email.it","ip":"127.0.0.1","azione":"login"}
[2025-02-14 14:48:49] local.DEBUG: From: CyberBlog <hello@example.com>
To: admin@theaulabpost.it
Subject: Nuova richiesta di lavoro ricevuta
MIME-Version: 1.0
Date: Fri, 14 Feb 2025 14:48:48 +0000
Message-ID: <0068c7257db62a883d0e7177bf4a957f@example.com>
Content-Type: text/html; charset=utf-8
Content-Transfer-Encoding: quoted-printable

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <h1>È arrivata una richiesta per il ruolo di revisor</h1>
  <p>Ricevuta da teresa@email.it</p>
  <h4>Messaggio:</h4>
  <p>&lt;p&gt;vorrei diventare revisor&nbsp;&lt;p&gt;</p></p>
</body>
</html>
[2025-02-14 14:50:20] local.INFO: Nuovo utente REVISOR {"user_id":8,"user_name":"teresa","role":"revisor","assigned_by":"Steve Lorren (Admin)","timestamp":"2025-02-14 14:50:20"}
```

```
[2025-02-17 15:29:34] local.WARNING: IP 127.0.0.1 has been blocked for 1 minute(s) due to too many attempts.
```

## Challenge 4: Manomissione input ➔ ssrf attack per api delle news

### Scenario:

Esiste la funzionalità di suggerimento “news recenti” in fase di scrittura dell'articolo per prendere ispirazione.

È presente un menu a scelta facilmente alterabile da ispezione elemento.

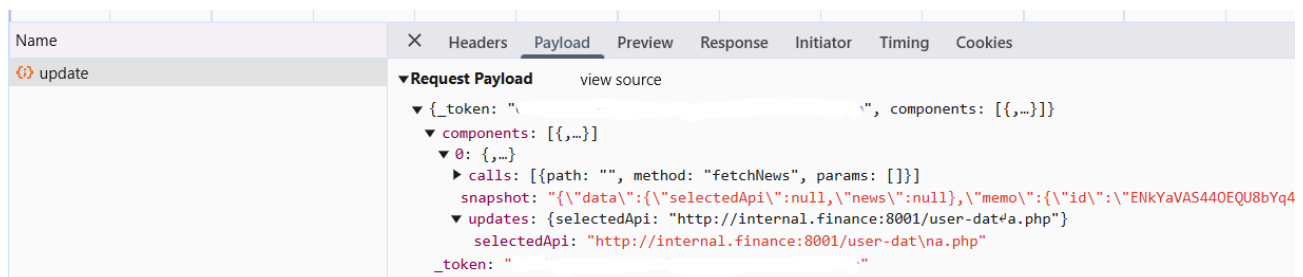
Nel file .env è presente la api key del servizio NewsApi, potete creare la vostra chiave e sostituirla.

### Attacco:

L'utente malintenzionato con un minimo di conoscenza del sistema cambia l'url e prova a far lanciare al server una richiesta che lui non sarebbe autorizzato (ssrf attack).  
Per esempio: il server recupera dei dati sugli utenti da un altro server in esecuzione sulla porta 8001.

### ➔ Prova iniziale dell'attacco SSRF:

```
<option value>Choose country</option>   
<option selected value="http://internal.finance:8001/user-data.php">NewsAPI - IT</option>   
<option value="https://newsapi.org/v2/top-headlines?country=gb&apiKey=5fbe92849d5648eabcbe072a1cf91473">NewsAPI - Uk</option> 
```



### Mitigazione:

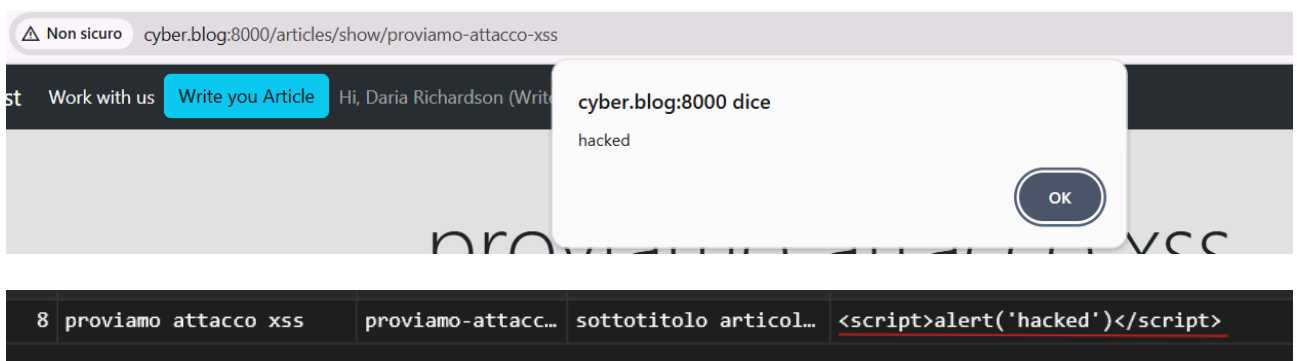
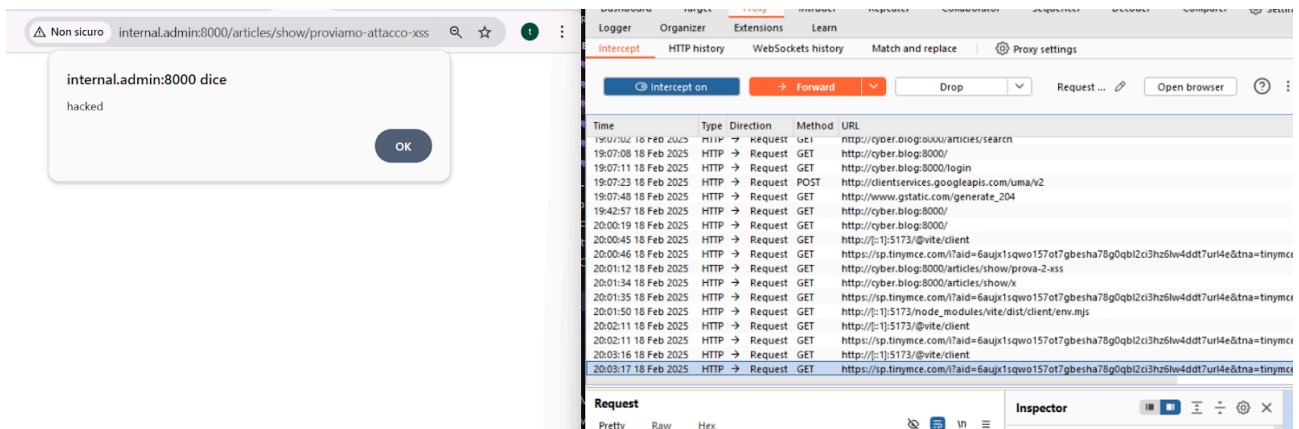
Rimodellare la funzionalità in modo tale da non poter lasciare spazio di modifica dell'url da parte di utenti malevoli. Implementare o migliorare la validazione degli input:

- Lavorare su App/Livewire/LatestNews.php migliorando la funzionalità, la validazione ecc..
- Lavorare su App/Services/HttpService.php giocando sui ruoli (Es. anche se è permesso fare request a internal.finance, se non sei admin ti dovrebbe bloccare, di solito l'hacker ha solo privilegi di writer).
- Inseriti gli url da cui sono consentite le richieste nella allowed\_origins in config/cors.php

### ➔ Prova dell'attacco SSRF dopo la mitigazione:

In network è possibile notare che non vi è più la richiesta 'update', che nell'attacco iniziale corrispondeva alla richiesta dell'utente malevolo.





## Mitigazione

Creare un meccanismo che filtri il testo prima di salvarlo e per essere sicuri anche in fase di visualizzazione dell'articolo.

### → Prova dell'attacco XSS dopo la mitigazione





## Challenge 6: Uso non corretto della proprietà fillable nei modelli

### Scenario

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.

Implementare una semplice pagina di profilo dell'utente (vista, rotte e controller), che preveda la modifica di:

- nome
- email
- password

Lasciamo la funzionalità appositamente vulnerabile per aumentare la consapevolezza sull'importanza della proprietà fillable nel contesto del mass assignment e di una buona validazione.

### ➔ Prova del Mass Assignment Attack prima della mitigazione:

id	name	email	is_admin	is_revisor	is_writer	email
1	Admin	admin@theaulabpost.it	1	0	0	NULL
2	Steven Manson	user@aulab.it	0	0	0	NULL

```
<input type="text" value="Steven Manson" id="name" class="form-control" name="name"> == $0  
<input type="hidden" value="1" id="is_admin" class="form-control" name="is_admin">
```

### Aggiorna il tuo profilo

Profilo aggiornato con successo.

Nome Utente:

id	name	email	is_admin	is_revisor	is_writer	email
1	Admin	admin@theaulabpost.it	1	0	0	NULL
2	Steven Manson	user@aulab.it	1	0	0	NULL

### Mitigazione

Nella proprietà fillable del modello in questione inserire solo i campi gestiti nel form.

➔ Prova del Mass Assignment Attack dopo la mitigazione:

id	name	email	is_admin	is_revisor	is_writer	
1	Admin	admin@theaulabpost.it	1	0	0	N
2	Steven Manson	user@aulab.it	0	0	0	N

```
▶ <label for="name" class="form-label">Ⓜ</label>  
<input type="text" value="Steven Manson" id="name" class="form-control" name="name">  
<input type="hidden" value="1" id="is_admin" class="form-control" name="is_admin"> == $0
```

id	name	email	is_admin	is_revisor	is_writer	
1	Admin	admin@theaulabpost.it	1	0	0	NU
2	Steven Manson	user@aulab.it	0	0	0	NU