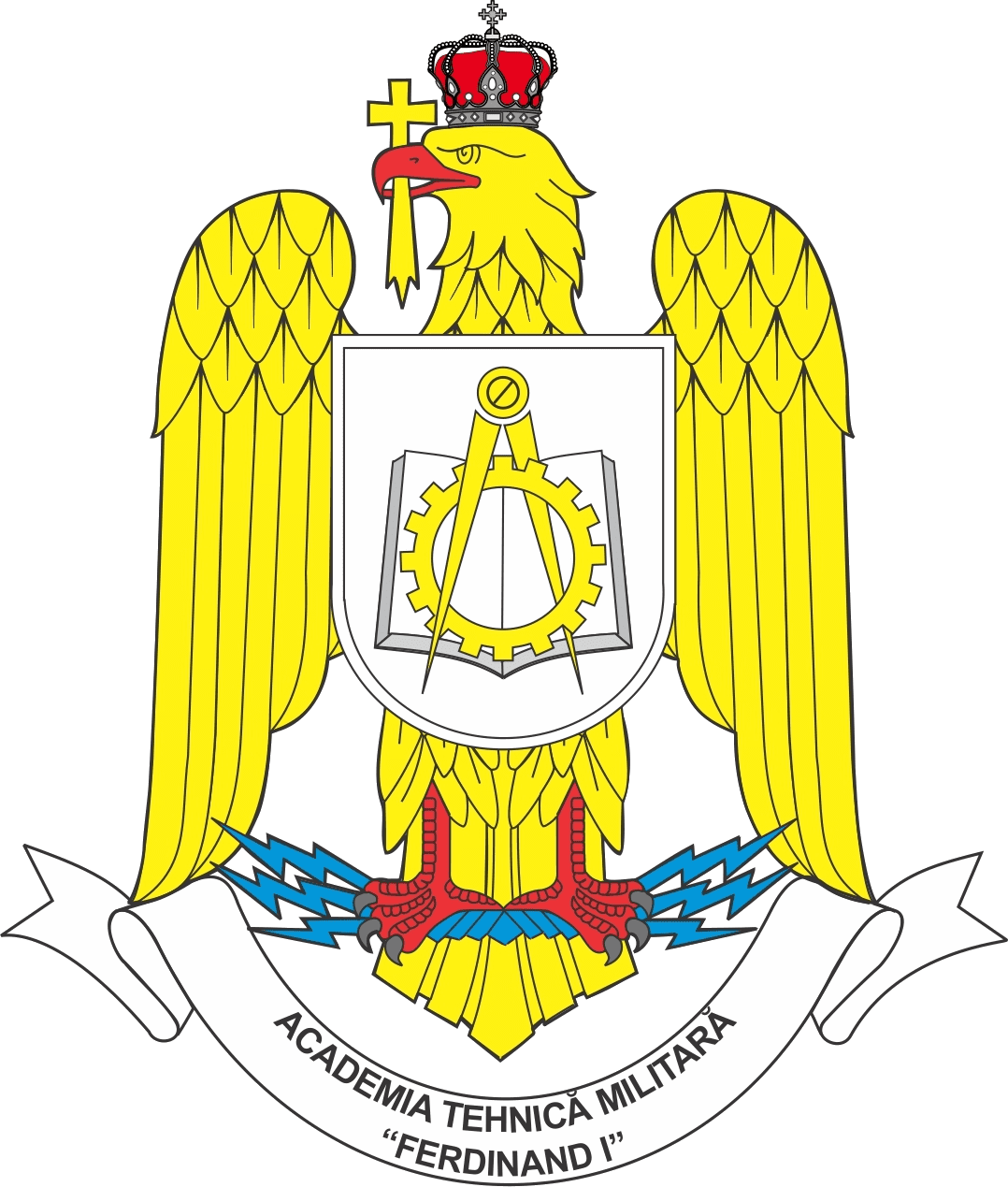
Academia Tehnică Militară „Ferdinand I”



PRACTICĂ 2023

Write up CTF-uri

STD. CAP. BĂRAR-PINTEA DANIEL-IOAN

GRUPA: C 112 C

Contents

[Forensic 3](#_Toc139011189)

[1. Deleted file 3](#_Toc139011190)

[2. Logs analysis - web attack 3](#_Toc139011191)

[Web-Client 6](#_Toc139011192)

[1. HTML – disabled buttons 6](#_Toc139011193)

[2. Javascript - Authentification 6](#_Toc139011194)

[3. Javascript - Source 7](#_Toc139011195)

[4. Javascript – Authentification 2 7](#_Toc139011196)

[5. Javascript – Obfuscation 1 7](#_Toc139011197)

[6. Javascript – Obfuscation 2 8](#_Toc139011198)

[Web-Server 9](#_Toc139011199)

[1. HTML – Source code 9](#_Toc139011200)

[2. HTTP – IP restriction bypass 9](#_Toc139011201)

[3. HTTP – Open redirect 10](#_Toc139011202)

[4. HTTP – User-agent 11](#_Toc139011203)

[5. Weak password 12](#_Toc139011204)

[6. PHP – Command injection 12](#_Toc139011205)

[7. Backup file 12](#_Toc139011206)

[8. HTTP - Directory indexing 13](#_Toc139011207)

[9. HTTP – Headers 15](#_Toc139011208)

[10. JWT – Introduction 16](#_Toc139011209)

[11. JWT - Revoked token 17](#_Toc139011210)

[12. JWT – Weak secret 21](#_Toc139011211)

[Reversing 24](#_Toc139011212)

[1. ELF x86 - 0 protection 24](#_Toc139011213)

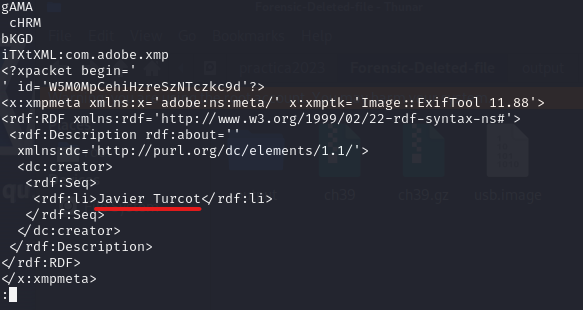
[2. ELF x86 – CrackPass 25](#_Toc139011214)

[3. ELF x86 - Anti-debug 29](#_Toc139011215)

# Forensic

## Deleted file

* Dezarhivez fisierul: tar -xf ch39.gz
* Folosesc comanda strings pentru a gasi patternuri de siruri de caractere : strings usb.imaga | less
* Am gasit numele proprietarului



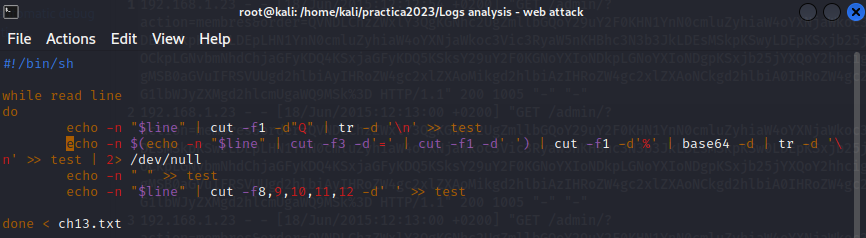
* Flag-ul este: javier\_turcot

## Logs analysis - web attack

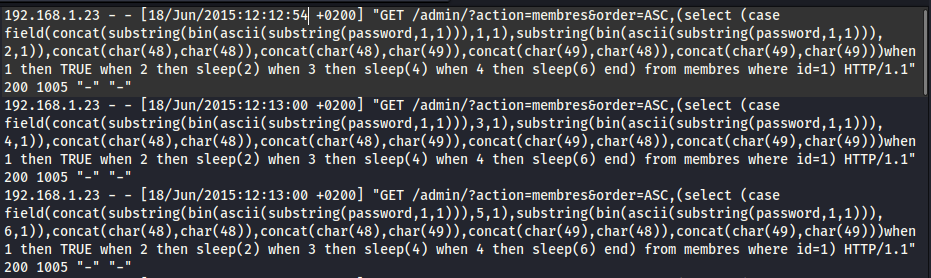
* Se observa ca cererile sunt codate in base64



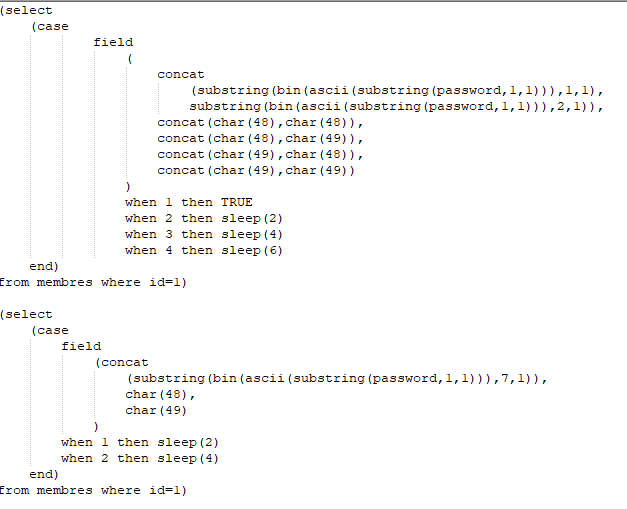
* Am facut un script bash pentru a decodifica mesajul



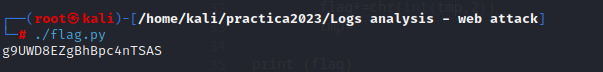
* In urma rularii scriptului am urmatoarele requesturi

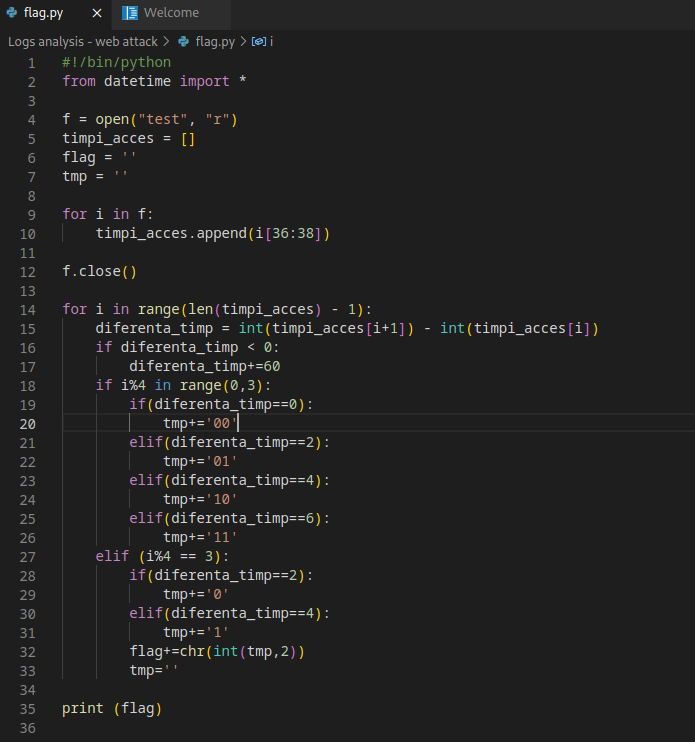


* Ce arata faptul ca este un atac de tipul sql injection asupra endpont-ului admin, atacatorul incearca sa afle parola prin compararea bitilor, pentru a fi mai lizibil am aliniat interogare:



* Se observa ca ia fiecare caracter pe rand (pana la 21), iar pentru fiecare caracter ia pe rand: bitii 1 si 2, bitii 3 si 4, bitii 5 si 6, bitul 7 (este nevoie de doar 7 biti pentru a reprezenta un caracter ascii), pentru fiecare pereche face un CASE, pentru fiecare potrivire se induce un delay de timp (in secunde) astfel atacatorul isi poate da seama ce biti sunt buni
* Am facut un script in pthon pentru a analiza diferentele de tip, astfel imi pot da seama ce biti sunt corecti, astfel incat sa ii tranform in cod ascii pentru a afla parola



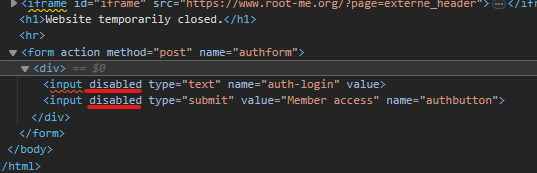


* Flag: g9UWD8EZgBhBpc4nTSAS

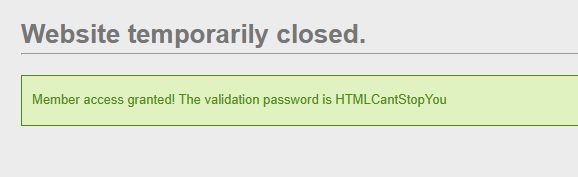
# Web-Client

## HTML – disabled buttons

* Inspect element pe pagina
* Sterg atributul disabled pentru inputuri



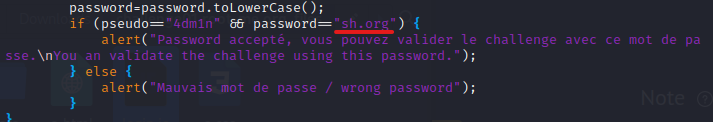
* Dupa ce introduc un text si apas butonul mi se deschide o pagina noua cu flag-ul



* FLAG: HTMLCantStopYou

## Javascript - Authentification

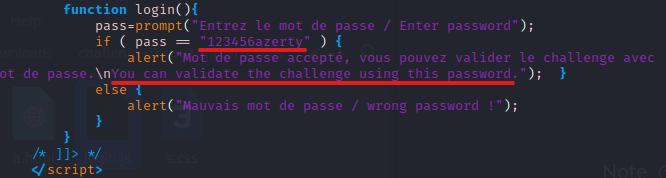
* Am descarcat codul paginii
* Am verificat fisierul login.js
* Am gasit flag-ul in credentialele de conectare

****

* Flag: sh.org

## Javascript - Source

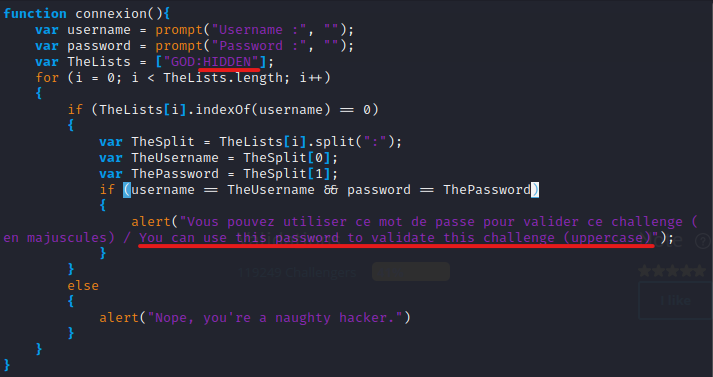
* Am descarcat sursa paginii
* Am deschis fisierul login.js
* Am gasit flag-ul la parola



* Flag: 123456azerty

## Javascript – Authentification 2

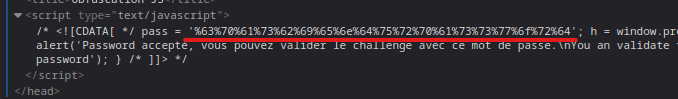
* Am descarcat codul sursa al paginii
* Am deschis fisierul login.js
* Am gasit flag-ul in codul din fisier



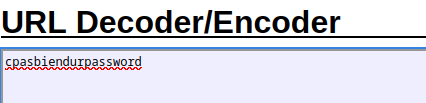
* Flag: HIDDEN

## Javascript – Obfuscation 1

* Am verificat codul sursa al paginii
* Am gasit o sintaxa codata URL



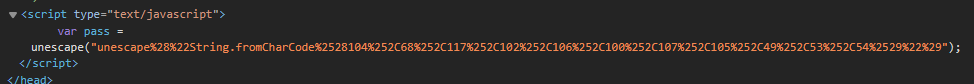
* Am decodificat sintaxa si am obtinut flag-ul



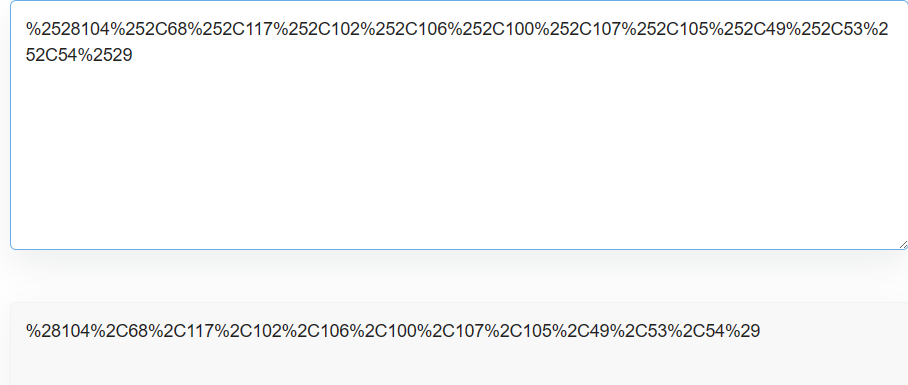
* Flag: cpasbiendurpassword

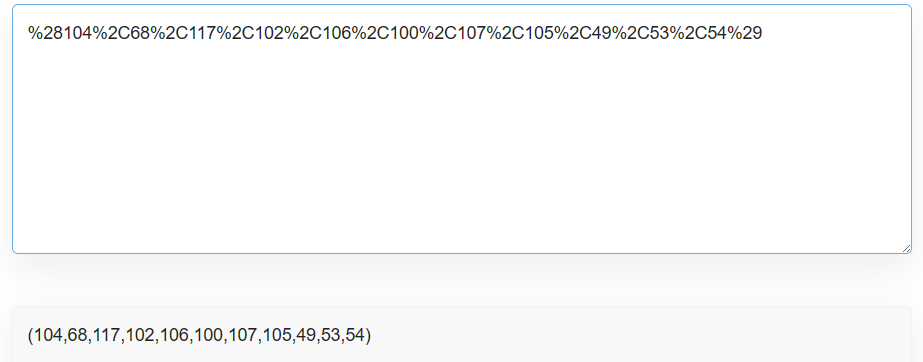
## Javascript – Obfuscation 2

* Am verificat sursa paginii si am gasit flag-ul escapat



* Pe urma am decodificat de doua ori pentru a gasi codul ASCII al flag-ului





* Am convertit aceste coduri ASCII pentru a obtine flag-ul

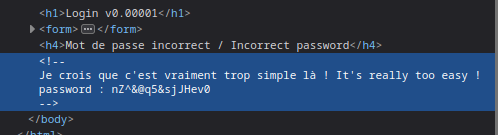


* Flag: hDufjdki156

# Web-Server

## HTML – Source code

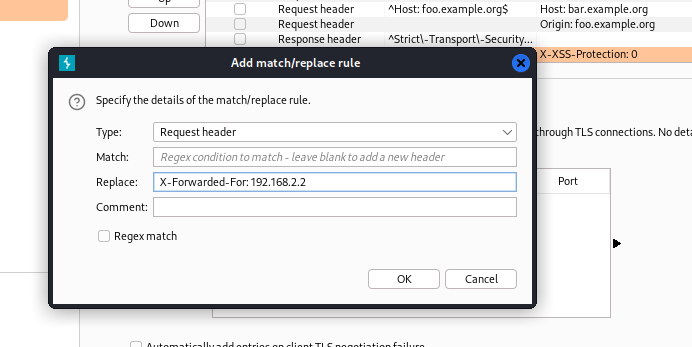
* Am verificat sursa paginii si am gasit flag-ul



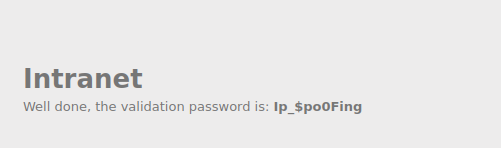
* Flag: nZ^&@q5&sjJHev0

## HTTP – IP restriction bypass

* Trebuie sa pacalesc serverul ca am o adresa locala pentru a accesa continutul
* Folosind Burp Suite am modificat header-ul ce contine adresa ip al solicitantului (in mod normal un proxy server adauga aceasta adresa ip)



* Pe urma am reincarcat pagina si am gasit flag-ul

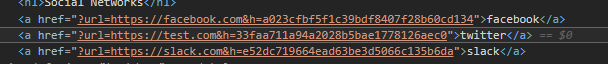


* Flag: **Ip\_$po0Fing**

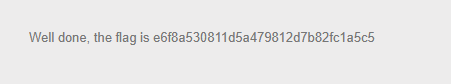
## HTTP – Open redirect

* Am citit resursa pusa la dispozitie si am aflat de vulnerabilitatea Open redirect si m-am folosit de ea
* Am modificat adresa de redirectare a twitter-ului si am adaugat si hash-ul sau (md5)





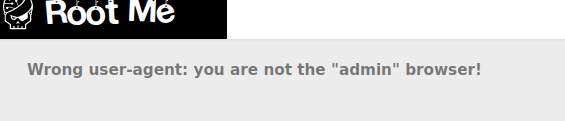
* Am accesat noul link si am gasit flag-ul



* Flag: e6f8a530811d5a479812d7b82fc1a5c5

## HTTP – User-agent

* Este nevoie sa modific user-agent-ul pentru a accesa continutul

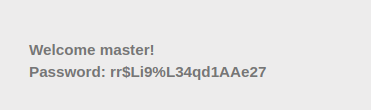


* Am folosit Burp suite pentru a intercepta traficul si modifica user-agent-ul din request





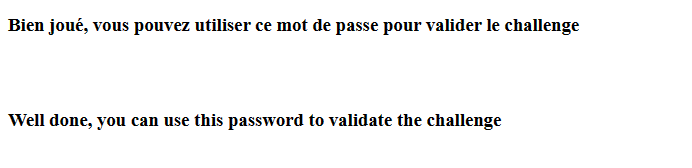
* A trebuit acum doar sa redirectionez request-ul catre server si am gasit flag-ul



* Flag: **rr$Li9%L34qd1AAe27**

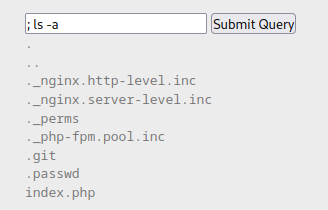
## Weak password

* La accesarea site-ului se cer credentialele adminului
* Am incercat forma cea mai comuna: user: admin , password: admin ; astfel am gasit flag-ul

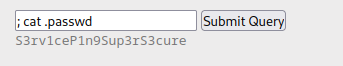
****

## PHP – Command injection

* Am obervat ca text input-ul paginii functioneaza ca si terminal de linux
* Este nevoie sa separ comanda introdusa prin semicoloana ( ; ) pentru a functiona
* Am cautat toate fisierele din directorul in care ma aflam



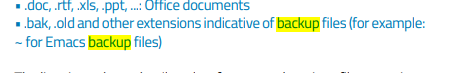
* Am gasit un fisier hidden cu nume intuitiv: .passwd
* Am afisat continutul fisierului si am gasit flag-ul



* Flag: S3rv1ceP1n9Sup3rS3cure

## Backup file

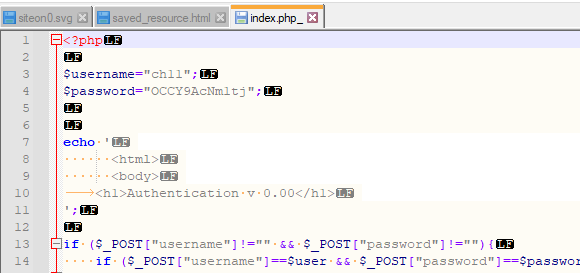
* Am cautat in resursele disponibile modul in care un site web php isi poate crea fisiere de backup



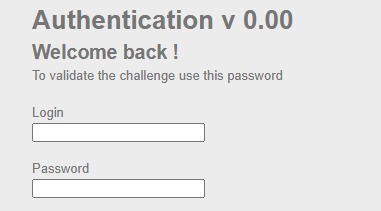
* Asadar am adaugat caracterul tilda la sfarsitul adresei URL pentru a accesa fisierul de backup



* Astfel s-a descarcat un fisier cu backup-ul siteului



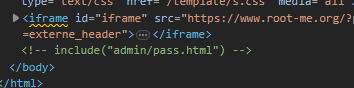
* Am introdus credentialele si astfel am afla flag-ul



* Flag: OCCY9AcNm1tj

## HTTP - Directory indexing

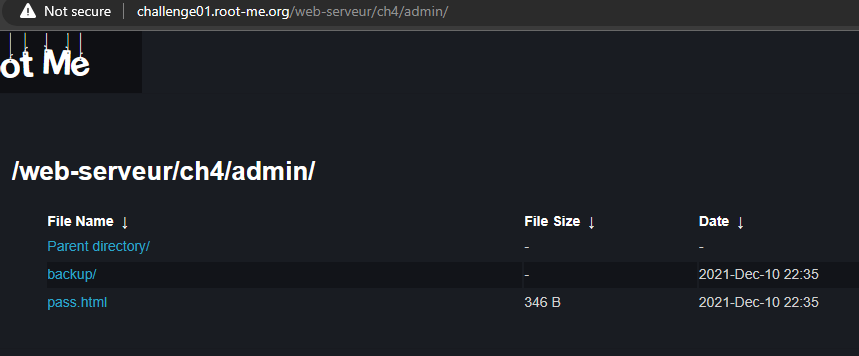
* Am verificat sursa paginii si am gasit o cale comentata



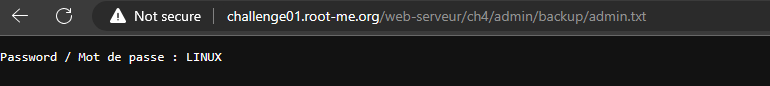
* Am adaugat calea la adresa URL, dar nu am gasit parola



* Asa ca am sters pass.html din cale pentru a vedea continutul fisierului admin



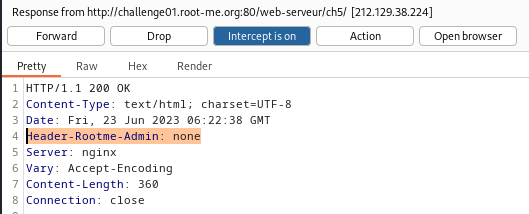
* Am gasit in directorul backup/admin.txt flag-ul cautat



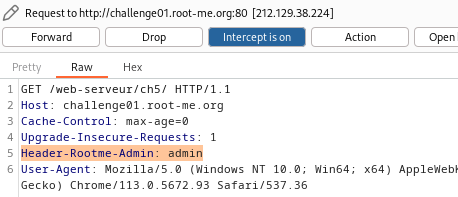
* Flag: LINUX

## HTTP – Headers

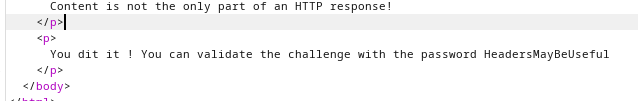
* Am folosit Burp Suite pentru a intercepta traficul cu serverul
* Am urmat hintul din challenge si m-am uitat pe headerele din request si response
* Am observat un header diferit la primul response



* I-am modificat valoarea si l-am introdus in primul request



* Astfel la urmatorul response am primit flag-ul



* Flag: HeadersMayBeUseful

## JWT – Introduction

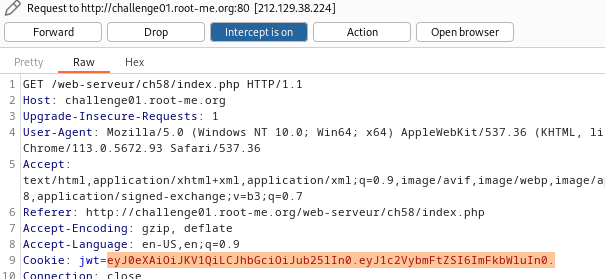
* Folosind Burp Suite am interceptat JWT-ul primit de la server



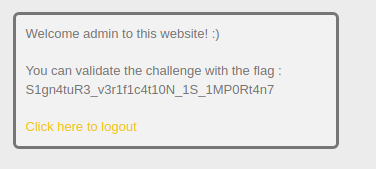
* Folosind un tool online ([JWT Debugger (token.dev)](https://token.dev/)) am modificat token-ul astfel incat sa am acces ca si admin, si sa nu se mai foloseasca nici un algoritm de verificare a token-ului



* Am modificat token-ul vechi cu cel nou in urmatorul request



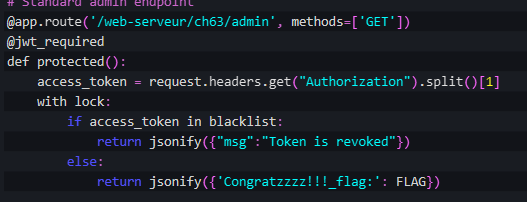
* Astfel am primit flag-ul



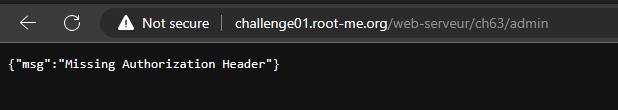
* Flag: S1gn4tuR3\_v3r1f1c4t10N\_1S\_1MP0Rt4n7

## JWT - Revoked token

* Am analizat scriptul server-ului si am observat urmatoarele aspecte:
* Trebuie sa accesez: <https://challenge01.root-me.org/web-serveur/ch63/admin> pentru a afla flag-ul



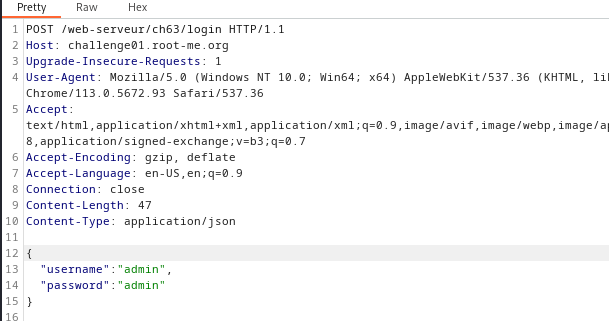
* Pentru a accesa acest endpoint am nevoie de Authorization Header



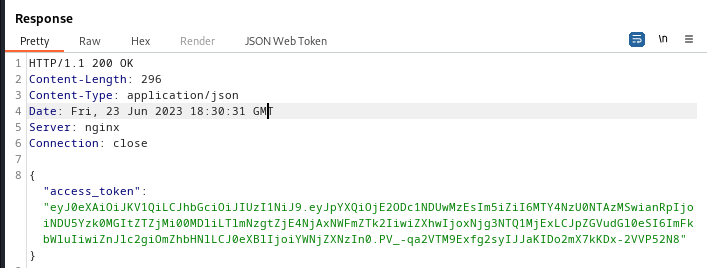
* Acest Authorization Header il gasesc in <https://challenge01.root-me.org/web-serveur/ch63/login>



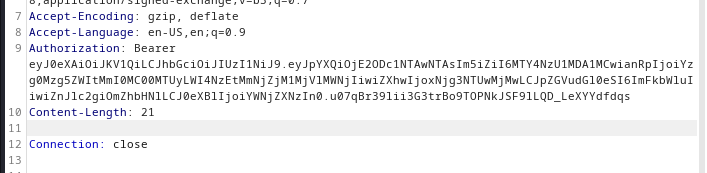
* Ca sa accesez acest endpoint este nevoie sa fac un request de tip POST in care sa adaug urmatorul json



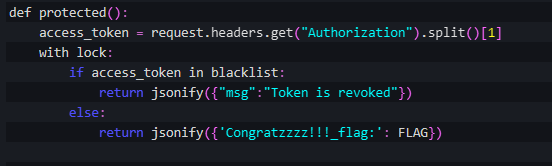
* Pentru a functiona acest request am adaugat un header in plus prin care specific layerul
* Dupa ce am dat forward catre site am access token-ul

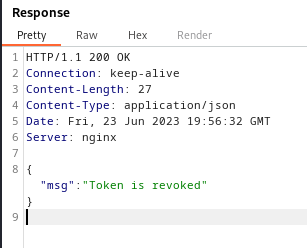


* Dupa ce am facut rost de token am adaugat un header nou la request-ul catre endpoint-ul admin

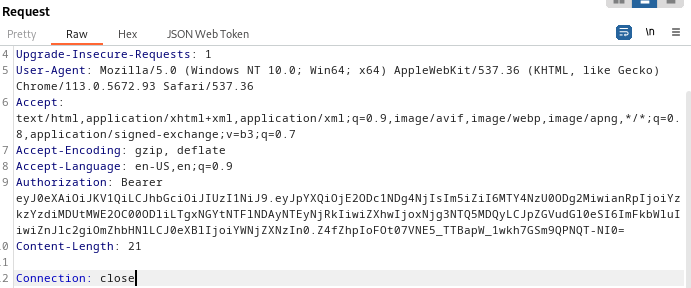


* Aceste server salveaza toate token-urile (timp de 3 minute) generate in endpoint-ul login si le salveaza in blacklist
* In momentul in care accesez endpoint-ul admin serverul verifica prezenta token-ului in blacklist si daca il gaseste atunci il va respinge

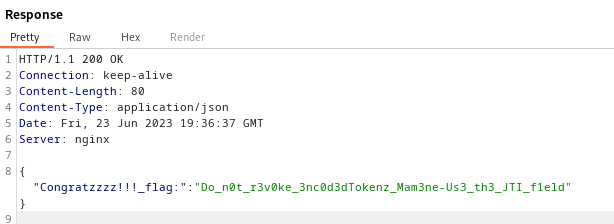




* O solutie pentru aceasta problema este sa adaug padding („=”) deoarece va fi diferit de primul token dar va avea aceasi valoare



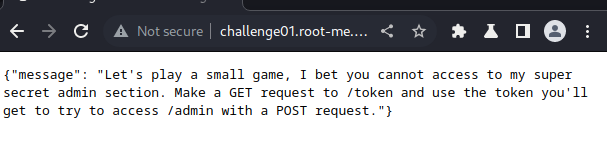
* In urma acestei modificari am dat forward catre server si am primit flag-ul



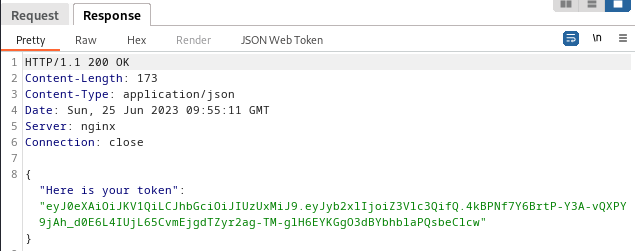
* Flag: Do\_n0t\_r3v0ke\_3nc0d3dTokenz\_Mam3ne-Us3\_th3\_JTI\_f1eld

## JWT – Weak secret

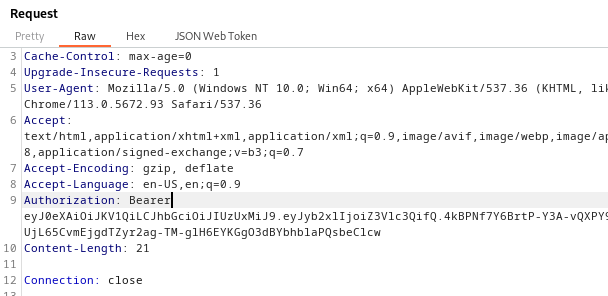
* Am urmat initial pasii din pagina principala



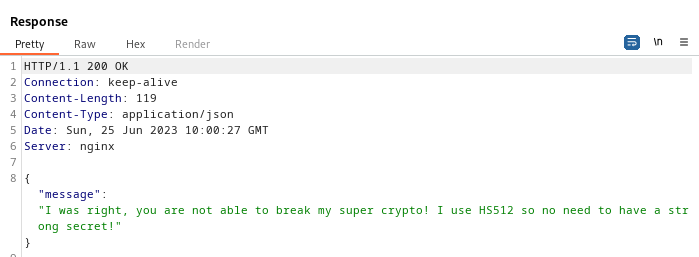
* Am facut un request de tip GET la endpoint-ul token si am facut rost de token



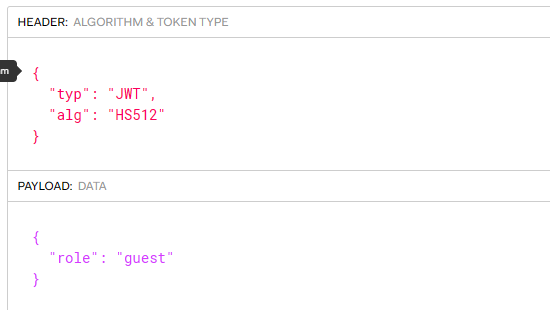
* Pe urma am facut un request de tip POST in care am adaugat si token-ul catre endpoint-ul admin ca sa obtin flag-ul



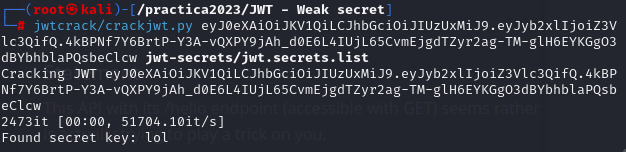
* Dar bineinteles token-ul nu era bun



* Token-ul are forma



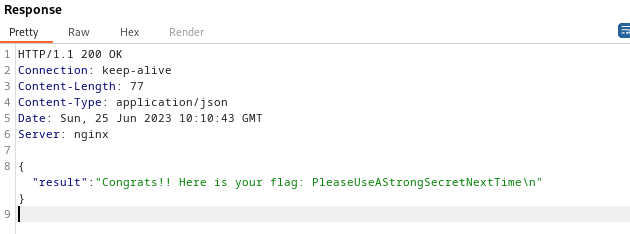
* Este nevoie sa modific parametrul de „role”, dar pentru asta aveam nevoie se secret
* Am folosit un tool de pe github ce foloseste brute force pentru a afla secretul ([Sjord/jwtcrack: Crack the shared secret of a HS256-signed JWT (github.com)](https://github.com/Sjord/jwtcrack)) si un dictionar cu secrete ([wallarm/jwt-secrets (github.com)](https://github.com/wallarm/jwt-secrets))



* In urma gasirii secretului a fost posibil sa modific JWT-ul



* Am facut din nou un request de tip POST catre endpoint-ul admin dar de data aceasta avand token-ul modificat
* Astfel am facut rost de flag

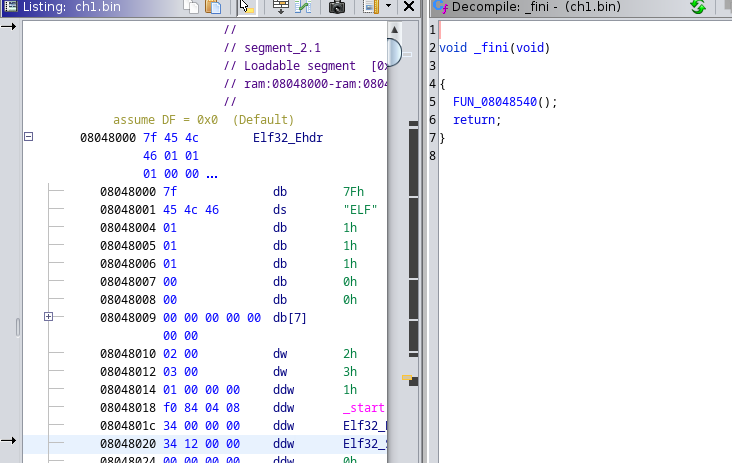


* Flag: PleaseUseAStrongSecretNextTime

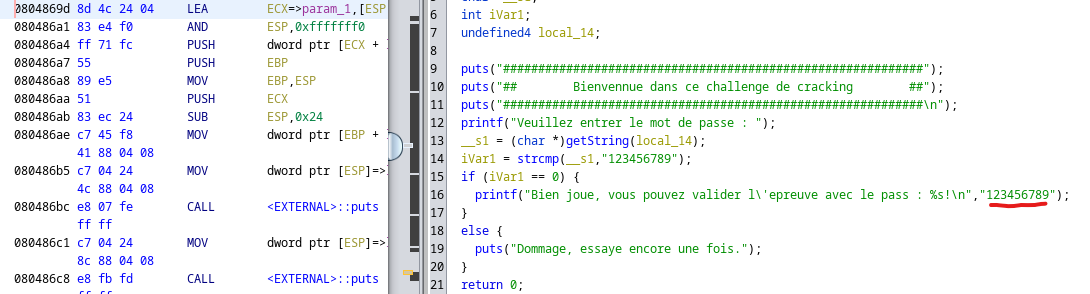
# Reversing

## ELF x86 - 0 protection

* Am folosit ghidra pentru a analiza codul compilat



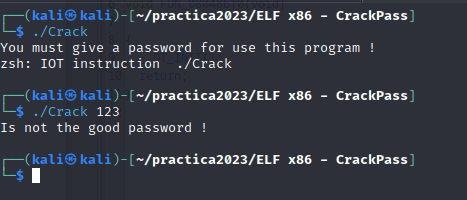
* Am verificat functiile pe rand si am gasit in functia main() flag-ul



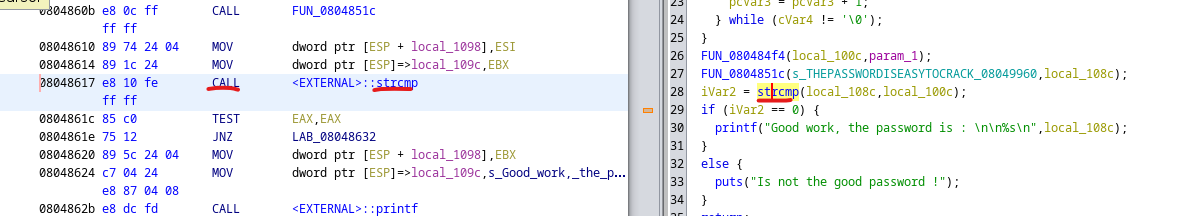
* Flag: 123456789

## ELF x86 – CrackPass

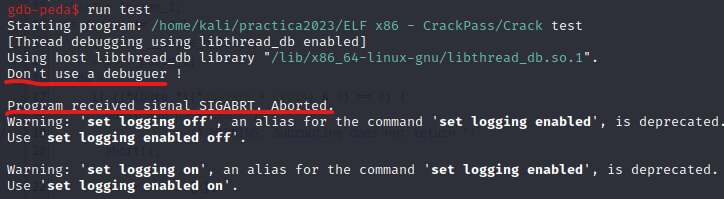
* Am incercat sa execut fisierul



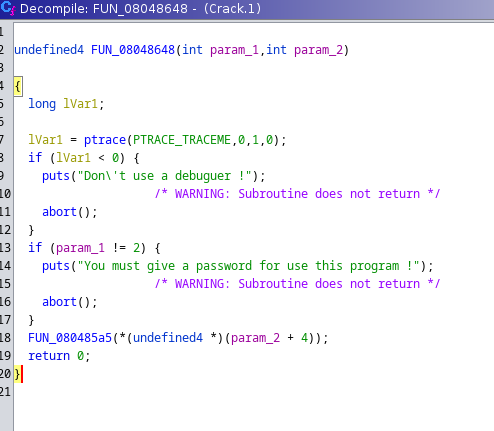
* Se observa ca am nevoie de o parola
* Folosind Ghidra am analizat codul dezasamblat
* Se observa ca mesajul de afisare a parolei depinde de un strcmp



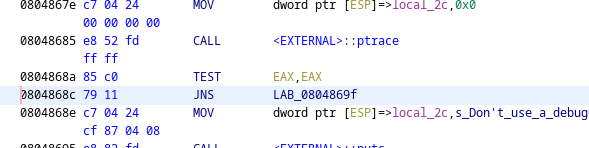
* Folosind gdb am incercat sa depanez sursa pentru a vedea ce valori va compara strcmp, dar sursa contine o metoda de protejare anti-debugging

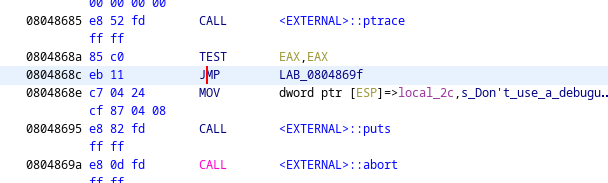


* Am cautat in codul sursa acest mesaj si am gasit urmatoarea functie

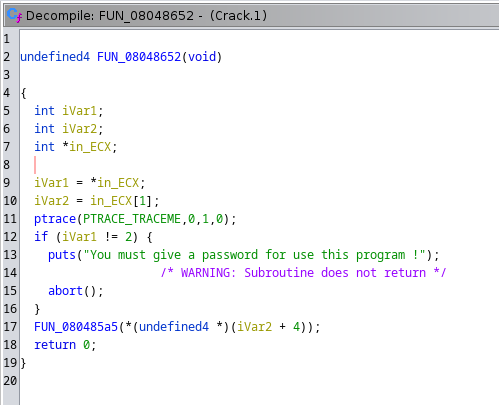


* Se foloseste ptrace(0,0,1,0) , aceasta detecteaza daca se foloseste modul debug pentru a se analiza codul
* Am analizat codul asm al acestei functii si am observat ca se poate „pacali” aceasta sintaxa conditionala prin modificarea unui jump

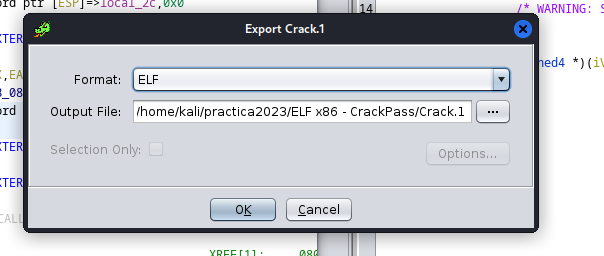




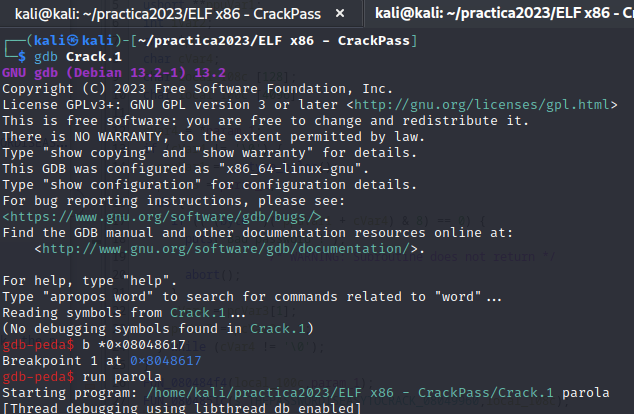
* Astfel obtin functia modificata:

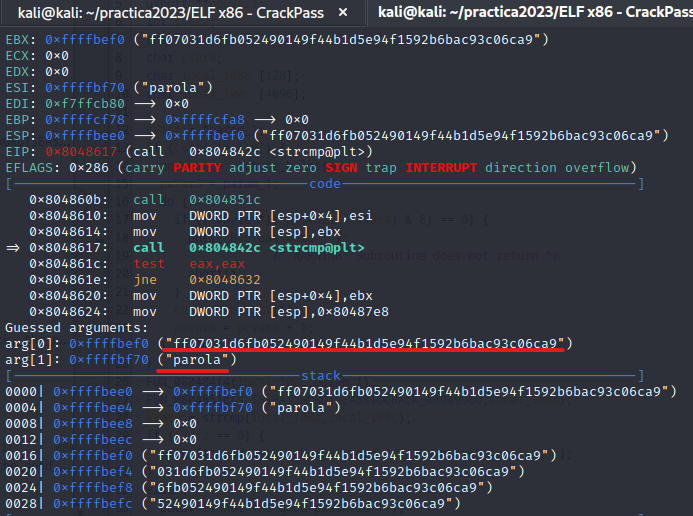


* Salvez noua sursa

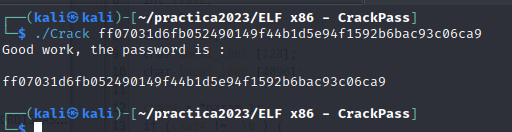


* Acum folosind utilitarul gdb pot face depanare pe codul sursa





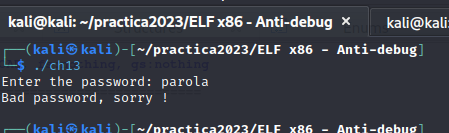
* Se face strcmp intre parola introdusa de mine si ff07031d6fb052490149f44b1d5e94f1592b6bac93c06ca9
* Am rulat din nou codul folosind de data aceasta parola gasita si am gasit flag-ul



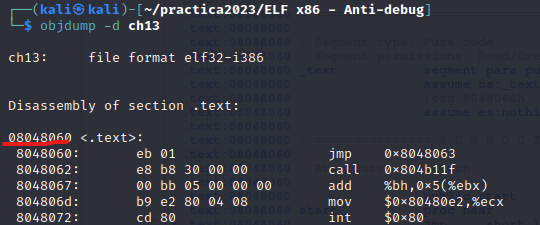
* Flag: ff07031d6fb052490149f44b1d5e94f1592b6bac93c06ca9

## ELF x86 - Anti-debug

* Am rulat scriptul dar imi cere o parola



* Este nevoie sa intru in modul debug pentru a analiza codul mai atent
* Pentru asta este nevoie sa vad unde incepe segementul de cod, am dat objdump pe fisier si am aflat ca adresa de inceput este 0x08048060



* Am facut debugging folosind gdb dar am intampinat un sigtrap in momentul in care s-a executat „int3”, care este un anti-debugger, se poate evita acesta debugger prin modificarea opcode-ului, int3 este pe un singur byte si are valoarea 0xCC, am modificat valoarea in 0x90 care reprezinta NOP (no operation)

