



ADDETC – Área Departamental de Engenharia Eletrónica e Telecomunicações  
e de Computadores

MEET – Mestrado Engenharia Eletrónica e Telecomunicações

MEIM - Mestrado Engenharia informática e multimédia

## **Cibersegurança**

### **Capture the Flag Challenge**

**Turma:**

MEIM-11D, MEET-31D

**Trabalho realizado por:**

Carlos Costa      N°45231

Tiago Martins      N°45240

Miguel Távora      N°45102

Duarte Domingues      N°45140

**Docente:**

Nuno Cruz

**Data:** 22/1/2022



# Índice

<b>1. INTRODUÇÃO .....</b>	<b>1</b>
<b>2. DESENVOLVIMENTO .....</b>	<b>2</b>
DESAFIO #1 – WIRESHARK DOO DOOO DO DOO... .....	2
DESAFIO #2 – WIRESHARK TWOO TWOOO TWO TWO... .....	3
DESAFIO #3 – SHARK ON WIRE 1 .....	5
DESAFIO #4 – PICOBROWSER .....	6
DESAFIO #5 – PLUMBING .....	7
DESAFIO #6 – WHERE ARE THE ROBOTS .....	8
DESAFIO #7 - WEBNET0 .....	9
DESAFIO #8 - WEBNET1 .....	10
DESAFIO #9 - EXTENSIONS .....	12
DESAFIO #10 – SO META .....	13
<b>3. CONCLUSÕES .....</b>	<b>14</b>

## Índice ilustrações

Figura 1 - Filtragem por respostas do tipo HTTP 200 OK .....	2
Figura 2 - Descriptação da flag no CyberChef.....	3
Figura 3 - Filtro com os pedidos para os subdomínios de reddshrimphandherring .....	3
Figura 4 - Descodificação através de base 64 no CyberChef .....	4
Figura 5 - Quantidade de pacotes UDP na captura “capture.pcap” .....	5
Figura 6 - Conteúdo de um dos pacotes UDP .....	5
Figura 7 - Flag numa stream UDP .....	5
Figura 8 - Acesso ao website com o browser errado .....	6
Figura 9 - Flag escondida no resultado do comando “curl” .....	6
Figura 10 - Output inicial do programa .....	7
Figura 11 - Output do programa após filtragem .....	7
Figura 12 – Disallow obtido a partir de robots.txt .....	8
Figura 13 - Flag picoCTF obtida .....	8
Figura 14 - Adição da RSA key ao wireshark .....	9
Figura 15 - Obtenção do conteúdo descriptado.....	9
Figura 16 - Obtenção da flag .....	9
Figura 17 - Adição da key RSA ao Wireshark .....	10
Figura 18 - Obtenção do conteúdo descriptado.....	10
Figura 19 - Flag movida.....	11
Figura 20 - Obtenção da flag .....	11
Figura 21 - Inicio do ficheiro de texto .....	12
Figura 22 - Flag da resolução do desafio .....	12
Figura 23 - Imagem pico_img.png.....	13
Figura 24 - Flag encontrada nas propriedades da imagem.....	13

# 1. Introdução

Neste trabalho foram realizados 10 desafios do website *picoCTF* com o intuito de exercitar os conhecimentos adquiridos nas aulas de Ciber Segurança. Destes desafios, 5 estão enquadrados na temática das redes, e os restantes são de outros diversos temas.

No total foram obtidos 1900 pontos na plataforma pela resolução dos 10 desafios.

## 2. Desenvolvimento

Neste capítulo são apresentadas as resoluções dos diferentes desafios, tendo sido resolvidos 5 desafios da área de redes, e outros 5 desafios de outras diversas áreas.

### Desafio #1 – Wireshark doo dooo do doo...

Depois de descarregado o ficheiro *shark1.pcapng* com os pacotes provenientes de uma captura de rede utilizou-se a ferramenta Wireshark para a análise do tráfego forma a se encontrar o código da *flag* correto.

Verificou-se que na sua maioria os pacotes existentes eram de HTTP e TCP.

Começou-se por analisar o *flow* dos pacotes HTTP do tipo *response* com o estado 200 OK.

Para isso utilizou-se o seguinte filtro *http.response.code==200* de acordo com a Figura 1:

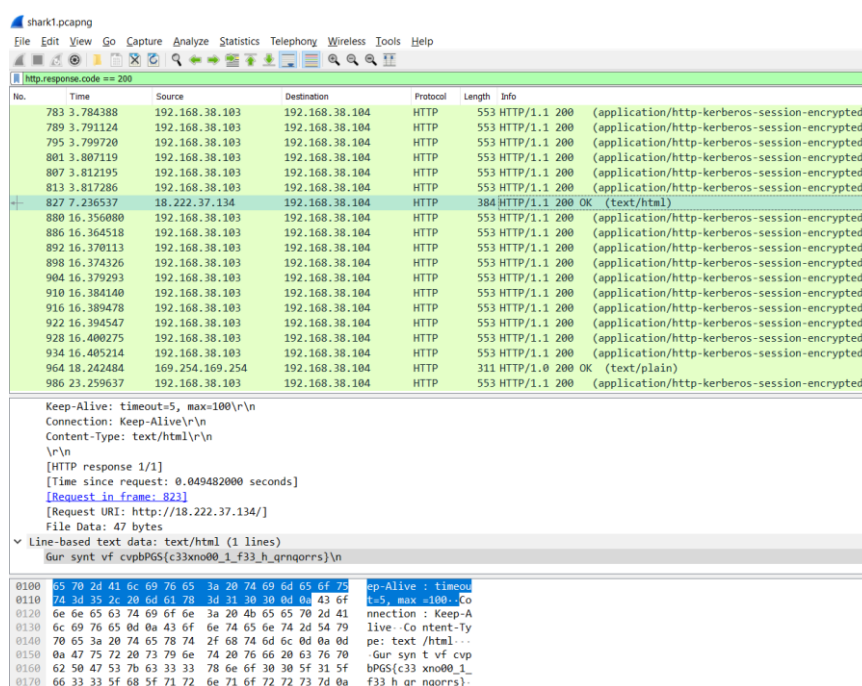


Figura 1 - Filtragem por respostas do tipo HTTP 200 OK

Numa destas *stream* verificou-se um formato de *flag* muito semelhante ao pretendido mas encriptado:

*Gur synt vf cvpbPGS{c33xno00\_1\_f33\_h\_qrnqorrs}\n*

Utilizou-se a ferramenta CyberChef na tentativa de decifrar a informação. A imagem da Figura 2 mostra o sucesso no processo de descriptação da *flag* através de força bruta da cifra de César que é uma cifra substituição.

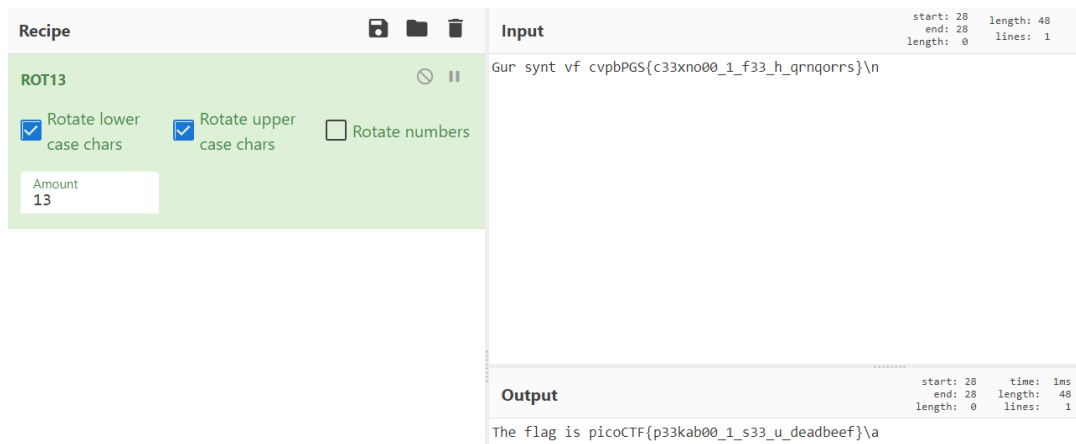


Figura 2 - Descriptação da flag no CyberChef

## Desafio #2 – Wireshark twoo twooo two twoo...

Começou-se por descarregar o ficheiro *shark2.pcapng* com o conteúdo relativo a uma captura de tráfego, onde se observou que a maioria dos pacotes registados estão associados aos protocolos HTTP e DNS.

Através da análise por *scan* da *stream* TCP, verificou-se que nos vários pedidos GET existiam diversas *flags* com o formato correto de acordo com o que se procurava. Não se tentou submeter essas flags porque se considerou que eram apenas uma distração.

De seguida, verificou-se que existiam diversos pedidos DNS para vários subdomínios de *reddshrimpan dherring.com* com o seguinte formato:

*XXXXXXXXX.reddshrimpan dherring.com*

Analizou-se melhor este tráfego por se considerar suspeito visto que a dica dada era procurar por tráfego deste tipo.

dns contains "local" and ip.dst!=8.8.8.8 and ip.src!=8.8.8.8 and dns.flags.response == 0						
No.	Time	Source	Destination	Protocol	Length	Info
1637	9.440363	192.168.38.104	18.217.1.57	DNS	109	Standard query 0x1dd2 A cG1jb0NU.reddshrimpan dherring.com.windowmain.local
2046	11.972605	192.168.38.104	18.217.1.57	DNS	109	Standard query 0xab99 A RntkbnNf.reddshrimpan dherring.com.windowmain.local
2448	14.605726	192.168.38.104	18.217.1.57	DNS	109	Standard query 0x9e21 A M3hm%Wxf.reddshrimpan dherring.com.windowmain.local
3153	16.506492	192.168.38.104	18.217.1.57	DNS	109	Standard query 0x2ee1 A ZnR3X2R1.reddshrimpan dherring.com.windowmain.local
3442	18.340155	192.168.38.104	18.217.1.57	DNS	109	Standard query 0x2a4b A YWriZiWm.reddshrimpan dherring.com.windowmain.local
3982	20.369626	192.168.38.104	18.217.1.57	DNS	105	Standard query 0x4068 A fQ==.reddshrimpan dherring.com.windowmain.local
4374	22.583745	192.168.38.104	18.217.1.57	DNS	105	Standard query 0x7418 A fQ==.reddshrimpan dherring.com.windowmain.local

Figura 3 - Filtro com os pedidos para os subdomínios de *reddshrimpan dherring*

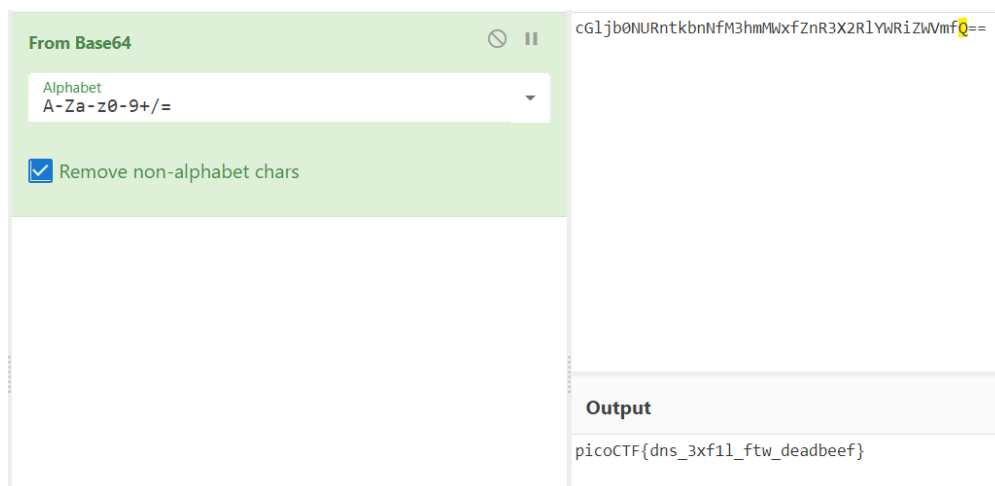
Para isso, de forma a filtrar o tráfego utilizou-se o seguinte comando apresentado na Figura 3 através do Wireshark.

Como existiam muitos pedidos para o servidor de DNS da google (8.8.8.8), decidiu-se ocultar esses mesmos pacotes, e após a filtragem obteve-se apenas aqueles para o endereço f18.217.1.57.

O próximo passo passou pela concatenação dos subdomínios tendo-se obtido a seguinte codificação:

*cGljb0NURntkbnNfM3hmMWxfZnR3X2RlYWUiZWVmfQ==*

Finalmente obteve-se a *flag* no formato correto através da ferramenta CyberChef pela decodificação de *string* obtida por base 64 como se pode observar na Figura 4.



**Figura 4 - Decodificação através de base 64 no CyberChef**



## Desafio #3 – shark on wire 1

Este desafio consiste em encontrar a *flag* numa captura de rede através do Wireshark. Analisando o ficheiro “capture.pcap” observou-se a existência de muitos pacotes UDP, como se pode observar na Figura 5, e quando se analisou o conteúdo de um destes pacotes (no campo “Data”) estes apenas continham o que aparentavam ser caracteres aleatórios, Figura 6.

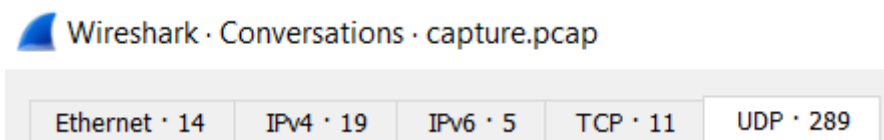


Figura 5 - Quantidade de pacotes UDP na captura “capture.pcap”

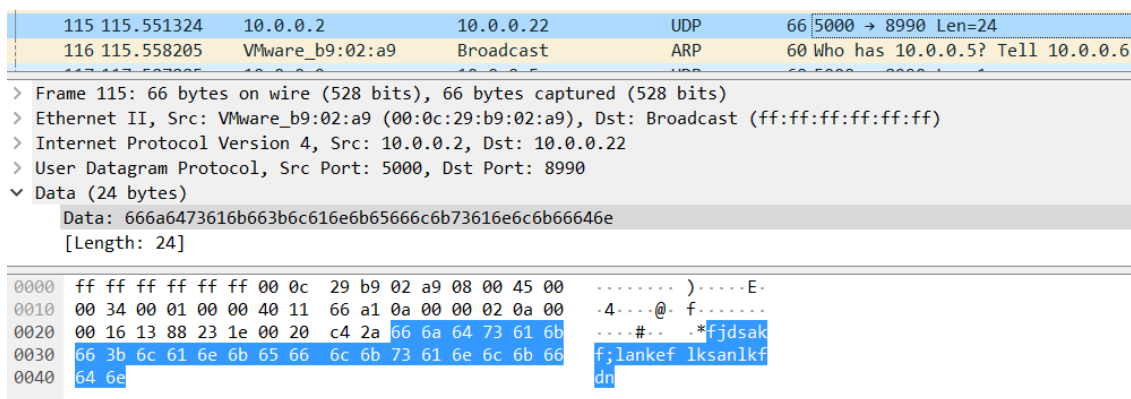


Figura 6 - Conteúdo de um dos pacotes UDP

Através da pista deste desafio, “What are streams?”, pensou-se que a *flag* estaria repartida entre diferentes pacotes UDP de uma certa conversa, ou seja, esta poderia ser encontrada analisando várias *streams* UDP. Desta forma foi usada a ferramenta “Follow UDP Stream”, com a qual foi possível encontrar a *flag* na *stream* número 6.

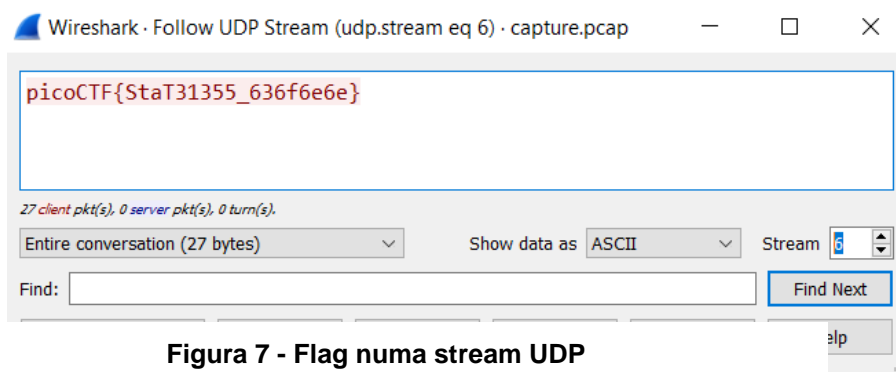


Figura 7 - Flag numa stream UDP

## Desafio #4 – picobrowser

Este desafio consiste em encontrar a *flag* num *website*, no entanto, esta apenas pode ser acedida se o acesso ao mesmo for realizado através do “picobrowser”. Inicialmente, quando se acedeu ao site, apareceu um erro a indicar que não acedemos com o *browser* correto.

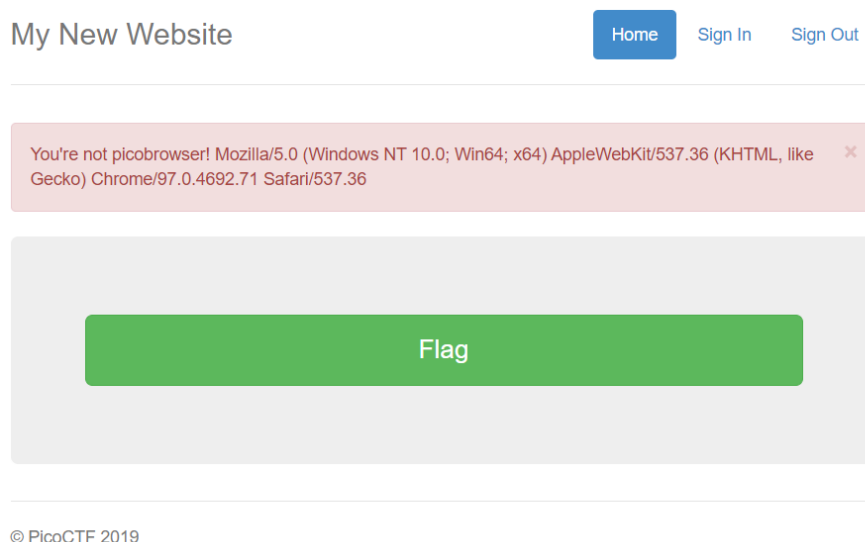


Figura 8 - Acesso ao website com o browser errado

Desta forma seria necessário aceder aos conteúdos deste *website* de uma forma alternativa que permitisse “enganá-lo” de forma que este pensasse que estávamos a realizá-lo através do picobrowser. Foi usada a ferramenta “curl” da linha de comandos com recurso à extensão “-A” que é responsável por alterar o *User-Agent* do pedido HTTP.

Assim, foi executado o seguinte comando, “curl -A “picobrowser” <https://jupiter.challenges.picoctf.org/problem/28921/flag>”, e na resposta do mesmo foi possível encontrar a *flag*.

```
<div class="jumbotron">
  <p class="lead"></p>
  <p style="text-align:center; font-size:30px;"><b>Flag</b>: <code>picoCTF{p1c0_s3cr3t_ag3nt_84f9c865}</code></p>
  <!-- <p><a class="btn btn-lg btn-success" href="admin" role="button">Click here for the flag!</a> -->
  <!-- </p> -->
</div>
```

Figura 9 - Flag escondida no resultado do comando “curl”

## Desafio #5 – plumbing

Este desafio consiste em encontrar a *flag* no *output* de um do site “jupiter.challenges.picoctf.org”, no porto 4427. Inicialmente foi realizado o acesso através da ferramenta “curl”, no entanto o output rapidamente ficou cheio de mensagens, onde aparentemente não estava presente a *flag*.

```
tiagofsmartins@LAPTOP-2UPSBR55:/mnt/c/Users/tiago$ curl jupiter.challenges.picoctf.org:4427
I don't think this is a flag either
This is defintely not a flag
Not a flag either
Not a flag either
This is defintely not a flag
I don't think this is a flag either
Again, I really don't think this is a flag
Not a flag either
Not a flag either
This is defintely not a flag
Again, I really don't think this is a flag
Again, I really don't think this is a flag
This is defintely not a flag
Not a flag either
```

Figura 10 - Output inicial do programa

Foi entanto decidido filtrar o resultado deste output para apenas incluir linhas que possuísem a palavra “picoCTF”. Para tal foi usado o comando “grep “picoCTF””, e este teve de ser concatenado ao comando “curl”, tendo sido usado para tal o comando “|”, também conhecido como *pipe* (referência ao título deste desafio).

Desta forma, foi possível obter a *flag* deste desafio, uma vez que foi o único resultado presente no *output* do programa.

```
tiagofsmartins@LAPTOP-2UPSBR55:/mnt/c/Users/tiago$ curl jupiter.challenges.picoctf.org:4427 | grep "picoCTF"
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left   Speed
100 14316  0 14316    0     0  34832    0 --:--:-- --:--:-- --:--:-- 34747picoCTF{digital_plumb3r_5ea1fbd7}
100 81476  0 81476    0     0  112k    0 --:--:-- --:--:-- --:--:-- 112k
```

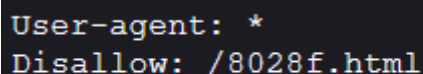
Figura 11 - Output do programa após filtragem

## Desafio #6 – Where are the robots

Este desafio consiste em encontrar a *flag* no site <https://jupiter.challenges.picoctf.org/problem/60915/>.

Essencialmente para conseguir acesso a informação acedemos ao ficheiro robots.txt, isto é feito por inserir no fim do site robots.txt, como se segue: <https://jupiter.challenges.picoctf.org/problem/60915/robots.txt>

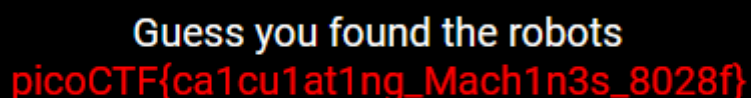
Este ficheiro serve essencialmente para controlar os ficheiros que os rastreadores podem ter acesso. Possui uma ou mais restrições, cada restrição bloqueia ou permite um rastreador de ter acesso ou não a ficheiros de um determinado caminho no website. Caso não seja especificado todos os ficheiros são permitidos para os rastreadores.



```
User-agent: *  
Disallow: /8028f.html
```

Figura 12 – Disallow obtido a partir de robots.txt

A partir deste ficheiro foi possível obter o ficheiro html e assim obter informação privilegiada. <https://jupiter.challenges.picoctf.org/problem/60915/8028f.html>



Guess you found the robots  
**picoCTF{ca1cu1at1ng\_Mach1n3s\_8028f}**

Figura 13 - Flag picoCTF obtida

## Desafio #7 - WebNet0

Este desafio consiste em decifrar a *stream* TLS, nessa mesma *stream* encontra-se a *flag*.

Para isto foi utilizado o Wireshark. Para obter as mensagens decifradas foi adicionado a *key* RSA no Wireshark, como se pode observar na imagem:

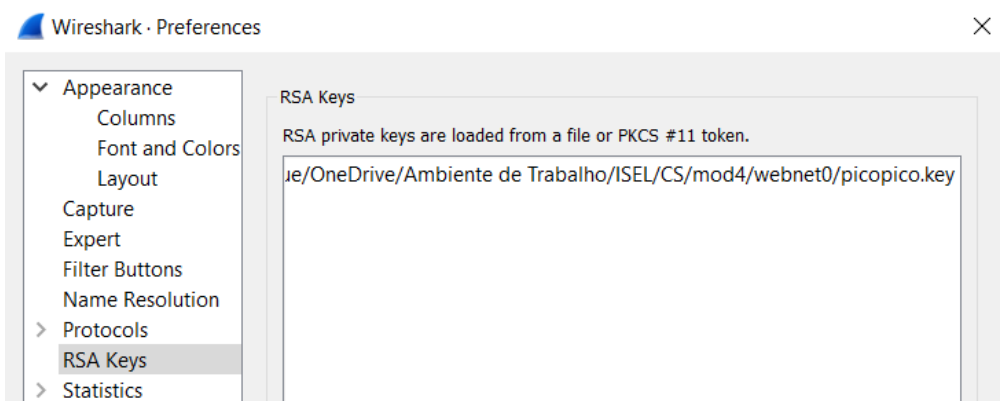


Figura 14 - Adição da RSA key ao wireshark

Após adicionar a *key*, foi reiniciado o Wireshark e obteve-se mensagens decifradas, representadas a amarelo na figura que se segue:

31	0.187804	128.237.140.23	172.31.22.220	HTTP	506 GET / HTTP/1.1
32	0.188303	172.31.22.220	128.237.140.23	HTTP	1299 HTTP/1.1 200 OK (text/html)
33	0.220287	128.237.140.23	172.31.22.220	TCP	66 57581 → 443 [ACK] Seq=1276 Ack=2499 Win=129792 Len=0 TSval=132865368 TSecr=568332936
34	0.357102	128.237.140.23	172.31.22.220	HTTP	521 GET /starter-template.css HTTP/1.1
35	0.357544	172.31.22.220	128.237.140.23	HTTP	576 HTTP/1.1 200 OK (text/css)
36	0.386999	128.237.140.23	172.31.22.220	TCP	66 57567 → 443 [ACK] Seq=1291 Ack=1776 Win=130560 Len=0 TSval=132865528 TSecr=568333105
37	0.817074	128.237.140.23	172.31.22.220	HTTP	438 GET /favicon.ico HTTP/1.1
38	0.817393	172.31.22.220	128.237.140.23	HTTP	637 HTTP/1.1 404 Not Found (text/html)

Figura 15 - Obtenção do conteúdo descriptado

Com a obtenção do conteúdo do HTML foi pesquisado no *header* do HTML e encontrou-se a chave.

31	0.187804	128.237.140.23	172.31.22.220	HTTP	506 GET / HTTP/1.1
32	0.188303	172.31.22.220	128.237.140.23	HTTP	1299 HTTP/1.1 200 OK (text/html)
33	0.220287	128.237.140.23	172.31.22.220	TCP	66 57581 → 443 [ACK] Seq=1276 Ack=2499 Win=129792 Len=0 TSval=132865368 TSecr=568332936

Hypertext Transfer Protocol					
HTTP/1.1 200 OK\r\n					
Date: Fri, 23 Aug 2019 15:56:36 GMT\r\n					
Server: Apache/2.4.29 (Ubuntu)\r\n					
Last-Modified: Mon, 12 Aug 2019 16:50:05 GMT\r\n					
ETag: "5ff-58fee50dc3fb0-gzip"\r\n					
Accept-Ranges: bytes\r\n					
Vary: Accept-Encoding\r\n					
Content-Encoding: gzip\r\n					
Pico-Flag: picoCTF{nongshim.shrimp.crackers}\r\n					
Content-Length: 821\r\n					

Figura 16 - Obtenção da flag

## Desafio #8 - WebNet1

Este desafio é semelhante ao WebNet0, mas desta vez a *flag* não se encontra na *header*.

Da mesma forma que anteriormente, foi utilizado o Wireshark. Começou-se também por obter as mensagens decifradas adicionando a RSA *key* no Wireshark, a partir do ficheiro “picopico.key”.

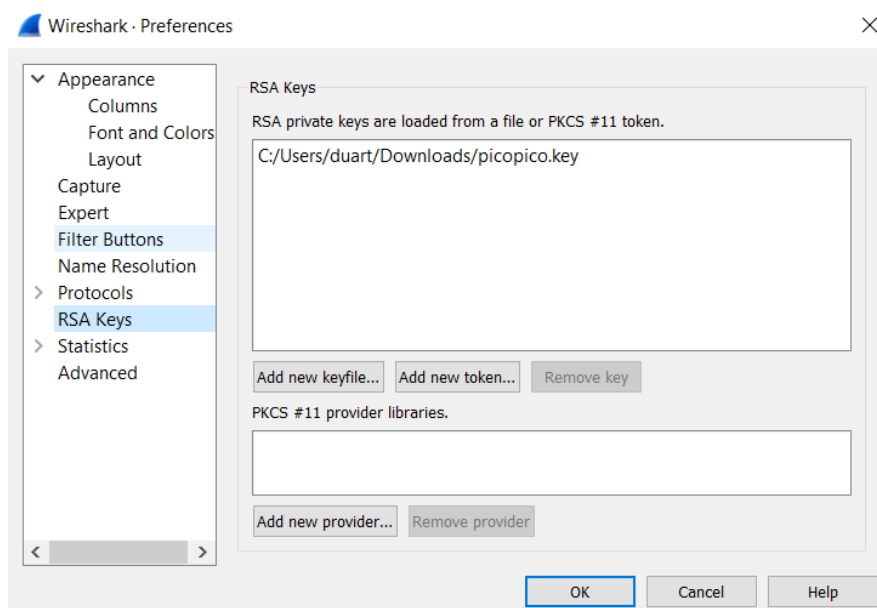


Figura 17 - Adição da key RSA ao Wireshark

Após adicionar a *key*, foi reiniciado o Wireshark e obteve-se mensagens HTTP decifradas, representadas a amarelo na figura que se segue:

No.	Time	Source	Destination	Protocol	Length	Info
109	0.799455	172.31.22.220	128.237.140.23	HTTP	637	HTTP/1.1 404 Not Found (text/html)
42	0.169756	172.31.22.220	128.237.140.23	HTTP	1330	HTTP/1.1 200 OK (text/html)
45	0.341085	172.31.22.220	128.237.140.23	HTTP	581	HTTP/1.1 200 OK (text/css)
91	0.611591	172.31.22.220	128.237.140.23	HTTP	2275	HTTP/1.1 200 OK (JPEG JFIF image)
47	0.550456	128.237.140.23	172.31.22.220	HTTP	519	GET /vulture.jpg HTTP/1.1
44	0.340614	128.237.140.23	172.31.22.220	HTTP	532	GET /starter-template.css HTTP/1.1
41	0.169256	128.237.140.23	172.31.22.220	HTTP	517	GET /second.html HTTP/1.1
108	0.799152	128.237.140.23	172.31.22.220	HTTP	438	GET /favicon.ico HTTP/1.1

Figura 18 - Obtenção do conteúdo descriptado

Com a obtenção do conteúdo do HTML foi pesquisado no *header* do HTML, como no desafio anterior, mas desta vez encontrou-se que a chave tinha sido movida.

No.	Time	Source	Destination	Protocol	Length	Info
109	0.799455	172.31.22.220	128.237.140.23	HTTP	637	HTTP/1.1 404 Not Found (text/html)
42	0.169756	172.31.22.220	128.237.140.23	HTTP	1330	HTTP/1.1 200 OK (text/html)
45	0.341085	172.31.22.220	128.237.140.23	HTTP	581	HTTP/1.1 200 OK (text/css)
91	0.611591	172.31.22.220	128.237.140.23	HTTP	2275	HTTP/1.1 200 OK (JPEG JFIF image)
47	0.550456	128.237.140.23	172.31.22.220	HTTP	519	GET /vulture.jpg HTTP/1.1
44	0.340614	128.237.140.23	172.31.22.220	HTTP	532	GET /starter-template.css HTTP/1.1
41	0.169256	128.237.140.23	172.31.22.220	HTTP	517	GET /second.html HTTP/1.1
108	0.799152	128.237.140.23	172.31.22.220	HTTP	438	GET /favicon.ico HTTP/1.1

> Frame 42: 1330 bytes on wire (10640 bits), 1330 bytes captured (10640 bits)  
 > Ethernet II, Src: 06:41:fb:0d:59:be (06:41:fb:0d:59:be), Dst: 06:72:25:91:2d:68 (06:72:25:91:2d:68)  
 > Internet Protocol Version 4, Src: 172.31.22.220, Dst: 128.237.140.23  
 > Transmission Control Protocol, Src Port: 443, Dst Port: 57944, Seq: 1266, Ack: 1287, Len: 1264  
 > Transport Layer Security  
 > Hypertext Transfer Protocol  
   > HTTP/1.1 200 OK\r\n
     Date: Fri, 23 Aug 2019 16:27:04 GMT\r\n
     Server: Apache/2.4.29 (Ubuntu)\r\n
     Last-Modified: Mon, 12 Aug 2019 17:35:36 GMT\r\n
     ETag: "624-58feef3af8698-gzip"\r\n
     Accept-Ranges: bytes\r\n
     Vary: Accept-Encoding\r\n
     Content-Encoding: gzip\r\n
     Pico-Flag: picoCTF{this.is.not.your.flag.anymore}\r\n
     Content-Length: 847\r\n

Figura 19 - Flag movida

Decidiu-se então seguir a *stream* HTTP (*Follow HTTP Stream*) desta forma é possível observar os dados a circular em entre cliente e servidor. Seguir uma HTTP *stream* permite decodificar informação HTTP, desta forma é possível observar o conteúdo das mensagens HTTP.

No.	Time	Source	Destination	Protocol	Length	Info
109	0.799455	172.31.22.220	128.237.140.23	HTTP	637	HTTP/1.1 404 Not Found (text/html)
42	0.169756	172.31.22.220	128.237.140.23	HTTP	1330	HTTP/1.1 200 OK (text/html)
45	0.341085	172.31.22.220	128.237.140.23	HTTP	581	HTTP/1.1 200 OK (text/css)
91	0.611591	172.31.22.220	128.237.140.23	HTTP	2275	HTTP/1.1 200 OK (JPEG JFIF image)

```

HTTP/1.1 200 OK
Date: Fri, 23 Aug 2019 16:27:04 GMT
Server: Apache/2.4.29 (Ubuntu)
Last-Modified: Fri, 23 Aug 2019 16:26:33 GMT
ETag: "112fb-590cb44f2cbe6"
Accept-Ranges: bytes
Content-Length: 70395
Pico-Flag: picoCTF{this.is.not.your.flag.anymore}
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: image/jpeg

.....JFIF.....Exif..MM.*.....J.....R.
(.....;.....Z.....picoCTF{honey.roasted.peanuts}.....ICC_PROFILE.....lcms.....mnrRGB XYZ
.....).9acspAPPL.....lcms.....
desc.....^cprt...
\...wptp...h...bkpt...|...rXYZ.....gXYZ.....bXYZ.....rTRC.....@gTRC.....@bTRC.....@desc.....c2.....
8.....XYZ.....b.....XYZ.....$......curv.....c...k...?.Q.4!.)..2.;.F.Qw].kpz....|.i.)...0....C.....
...
) /'%' /9339GDG11}..C.
  
```

Packet 91, 3 client pkts, 3 server pkts, 5 turns. Click to select.

Entire conversation (72 kB)

Find: pico

Show and save data as ASCII

Find Next

Filter Out This Stream Print Save as... Back Close Help

Figura 20 - Obtenção da flag

Realizou-se *Follow HTTP Stream* em um pacote de uma imagem (JPEG JFIF image) e procurou-se por pico, na barra de pesquisa. Encontrou-se finalmente a *flag* (picoCTF{honey.roasted.peanuts}) na secção dos dados da imagem.

## Desafio #9 - Extensions

O objetivo é descobrir a *flag* que se encontra num ficheiro txt. O ficheiro começava por %PNG, como se observa na imagem que se segue.

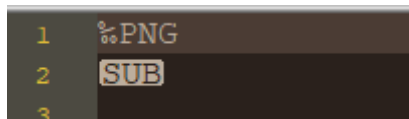


Figura 21 - Inicio do ficheiro de texto

Desta forma surgiu a questão de poder ser um ficheiro PNG, outra pista é o facto de o conteúdo estar escrito em bytes. Desta forma o objetivo era converter o ficheiro para PNG. Foi utilizado o sistema operativo Windows, mas não era possível obter o conteúdo da imagem. Por isso foi utilizado um website de conversão de ficheiro texto para PNG, nomeadamente <https://products.groupdocs.app/conversion/txt-to-png>. O ficheiro obtido foi:

```
picoCTF{now_you_know_about_extensions}
```

Figura 22 - Flag da resolução do desafio



## Desafio #10 – So Meta

Este desafio consiste em encontrar a *flag* numa imagem png (pico\_img.png), o desafio pode ser encontrado em

<https://play.picoctf.org/practice/challenge/19?page=1&search=so%20meta>.



Figura 23 - Imagem pico\_img.png

Este exercício baseia-se em encontrar uma *flag* em uma imagem png, portanto teve-se a ideia de visualizar a *metadata* da imagem.

Utilizou-se o *magick identify*, que faz parte do *ImageMagick*, um programa que permite obter o formato e características de uma imagem.

Usou-se o comando **identify -verbose**, este comando permite imprimir informação detalhada da imagem.

```
Properties:
  Artist: picoCTF{s0_m3ta_eb36bf44}
  date:create: 2022-01-19T14:27:04+00:00
  date:modify: 2022-01-19T14:27:20+00:00
  png:IHDR.bit-depth-orig: 8
  png:IHDR.bit_depth: 8
  png:IHDR.color-type-orig: 2
  png:IHDR.color_type: 2 (Truecolor)
  png:IHDR.interlace_method: 0 (Not interlaced)
  png:IHDR.width,height: 600, 600
  png:sRGB: intent=0 (Perceptual Intent)
  png:text: 2 tEXt/zTXt/iTXt chunks were found
  signature: bd31af9e175e0c6b09e6632b7fb0821622f74a0487bea6511dbb26988540af4e
  Software: Adobe ImageReady
```

Figura 24 - Flag encontrada nas propriedades da imagem

Nas propriedades da imagem foi possível encontrar a *flag* picoCTF{s0\_m3ta\_eb36bf44} e concluir o desafio.

### **3. Conclusões**

Em suma, com este trabalho foi possível consolidar os conhecimentos adquiridos na unidade curricular e aplicá-los na resolução de desafios de diversas temas, maioritariamente na área de redes.