Instituto Superior de Engenharia de Lisboa

Cibersegurança

**Desafios Capture the Flag**

Mestrado em Engenharia Informática de Multimédia

Pedro Gonçalves, 45890

Rodrigo Dias, 45881

Rúben Santos, 49063

Semestre de Inverno, 2021/2022

# Introdução

Este projeto procurará solucionar alguns desafios disponibilizados pela plataforma **picoCTF** (***capture the flag***). Muitos dos desafios encaixam no âmbito de segurança na rede, e por isso utiliza-se recorrentemente software **Wireshark**.

# Desafios

## Wireshark Doo Dooo Do Doo…

Texto

Descrição gerada automaticamenteO primeiro desafio fornece um registo de pacotes capturados na rede (o ficheiro **shark1.pcapng**). Para abrir o ficheiro e analisar os pacotes, utilizar-se-á o software **Wireshark**.

Ao analisar os pacotes (por alto, ainda sem inspecionar os detalhes), imediatamente identificam-se algumas linhas suspeitas, como a **817** e **818**, que representam uma comunicação através do protocolo **ARP**. Tipicamente, encontram-se pacotes **ARP** quando se inicia uma conversação, visto que é através desse protocolo que os endereços são descobertos.

Ao procurar todos os pacotes **ARP**, verifica-se que apenas existem estes dois.

Texto, Carta

Descrição gerada automaticamenteAnalisando a ***stream*** (fluxo de pacotes de uma conversação na rede) que vem a seguir (linha **819**), descobre-se uma linha que tem um formato parecido com a flag que se pretende encontrar. Contudo parece protegida por algum tipo de cifra.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Descobre-se então que a cifra utilizada é a ROT-13, uma cifra de translação em que o alfabeto de encriptação se desloca 13 posições (portanto, metade do alfabeto latino básico, com 26 letras). Decifrando, a mensagem é a seguinte:

**“The flag is picoCTF{p33kab00\_1\_s33\_u\_deadbeef}”**

**Então, a flag é a seguinte:**

**“picoCTF{p33kab00\_1\_s33\_u\_deadbeef}”**

## Wireshark Twoo Twooo Two Twoo

Interface gráfica do usuário, Texto, Carta

Descrição gerada automaticamenteO segundo desafio fornece outro registo de pacotes capturados na rede (o ficheiro **shark2.pcapng**). Da mesma forma que o desafio anterior, se se inspecionarem os diferentes streams contidos no registo, verifica-se que existem várias flags escondidas da mesma forma, todas diferentes, e desta vez nem estão encriptadas.

Isto leva a crer que não passam de uma ratoeira. Por essa razão, é preferível investigar os pacotes um pouco mais. Verifica-se que existem vários pacotes DNS, todos com um domínio muito suspeito, com o formato:

**“xA3F0VZ8.reddshrimpandherring.com”**

Uma imagem contendo Padrão do plano de fundo

Descrição gerada automaticamenteEm que “**xA3F0VZ8”** assume diferentes valores.

Tabela

Descrição gerada automaticamenteVerifica-se também que estes pacotes **DNS** ocorrem sempre numa conversação entre o endereço **192.168.38.104** e outro endereço. Se se filtrarem os pacotes, de forma a visualizar apenas os DNS e aqueles que contêm o endereço **192.168.38.104** como IP de destino, observa-se o seguinte:

Tabela

Descrição gerada automaticamenteExistem vários pacotes com o endereço de origem **8.8.8.8**, que é o serviço de **DNS** público disponibilizado pela **Google**. Desprezando esses, e ordenando:

Obtém-se então uma lista mais limitada. A ideia será concatenar os prefixos do domínio **“reddshrimpandherring.com”** obtendo aquilo que se suspeita tratar de uma mensagem cifrada. O resultado da concatenação é:

**“cGljb0NURntkbnNfM3hmMWxfZnR3X2RlYWRiZWVmfQ==”**

Ora, é sabido que as mensagens que utilizam o ***encoder*** **base64** utilizam **“==”** como um sufixo especial. Descodificando:

**“picoCTF{dns\_3xf1l\_ftw\_deadbeef}”**

## WPA-ing Out

Para este terceiro desafio, sabe-se que a ***flag*** secreta é uma password contida na wordlist **Rockyou.txt**. Utilizando a ferramenta **Aircrack-ng**, torna-se muito fácil:

Texto

Descrição gerada automaticamente

A flag é:

**“picoCTF{mickeymouse}”**

## Shark on Wire 1

Tabela

Descrição gerada automaticamenteDesta vez, é fornecido um ficheiro **capture.pcap**, que será também analisado no **Wireshark**. Este ficheiro conta com a captura de centenas de pacotes **UDP**, que não parecem conter qualquer informação que aponte na direção da ***flag***.

Contudo, ao analisar com mais atenção, verifica-se que alguns pacotes suspeitos, nomeadamente os que têm um IP de destino, por exemplo, igual a **10.0.0.13** ou a **10.0.0.12**, apresentam um campo **“data”** com dimensão de **1** **byte**.

Tabela

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Ao filtrar os pacotes, de forma a visualizar apenas os que têm endereço de destino **10.0.0.13**, obtém-se a seguinte lista.

Tabela

Descrição gerada automaticamente

Concatenando os valores (em hexadecimal) dos bytes dos vários pacotes, obtém-se:

**7069636f4354467b4e30745f615f664c61677d**

Convertendo para caracteres **ASCII**:

**picoCTF{N0t\_a\_fLag}**

Trata-se de uma ratoeira. Contudo, se o mesmo procedimento for realizado, mas para pacotes com o endereço de destino **10.0.0.12**, obtém-se:

**7069636f4354467b5374615433313335355f36333666366536657d**

Convertendo para caracteres **ASCII**, encontramos finalmente a **flag** secreta:

**picoCTF{StaT31355\_636f6e6e}**

## Shark on Wire 2

Este desafio é semelhante ao anterior. É novamente fornecido um ficheiro **capture.pcap**, que será mais uma vez analisado no **Wireshark**. Da mesma forma que o desafio anterior, este ficheiro conta com a captura de centenas de pacotes **UDP**.

Texto

Descrição gerada automaticamenteTexto

Descrição gerada automaticamente com confiança médiaAo analisar as ***streams*** **UDP**, encontram-se diversas mensagens muito interessantes, mas nenhuma aponta na direção da verdadeira **flag**.

Com exceção de duas mensagens:





Após a primeira mensagem, que parece indicar o começo de algo, e antes da segunda mensagem, que parece indicar o final de algo, existem várias mensagens vindas do **IP** de origem **10.0.0.66**. Filtrando os pacotes com origem nesse endereço, obtém-se:

Tabela

Descrição gerada automaticamente

Destaca-se imediatamente o facto de todos os pacotes terem diferentes portos associados. Sendo que todos começam com o algarismo **5**, tomar-se-á nota dos três restantes algarismos de cada porto, para, posteriormente, converter tudo para caracteres **ASCII**, tal como foi realizado no desafio anterior.

**000 112 105 099 111 067 084 070 123 112 049 076 076 102 051 114 051 100 095 100 097 116 097 095 118 049 097 095 115 116 051 103 048 125**

Convertendo para caracteres **ASCII**:

**picoCTF{p1LLf3r3d\_data\_v1a\_st3g0}**

## Lets Warm Up

Fez-se este desafio para preparar para a resolução do desafio **Nice netcat...** e consiste simplesmente em ver qual o símbolo ASCII corresponde ao valor hexadecimal 0x70. Como tal, visualizando numa tabela ASCII, o símbolo correspondente é a letra **p**.

**picoCTF{p}**

## what's a net cat?

Este desafio serviu também para entender como fazer o desafio **Nice netcat…** e para o resolver, foi apenas necessário abrir um terminal Linux para correr o comando **nc**, abrindo uma conexão com **jupiter.challenges.picoctf.org** no porto **41120**, obtendo a flag:

**picoCTF{nEtCat\_Mast3ry\_3214be47}**

## Nice netcat…

Para este desafio foi pedido executar o comando **nc mercury.picoctf.net 49039** num terminal Linux e obteve-se a seguinte sequência de números inteiros:

**112 105 099 111 067 084 070 123 103 048 048 100 095 107 049 116 116 121 033 095 110 049 099 051 095 107 049 116 116 121 033 095 051 100 056 052 101 100 099 056 125**

Assumiu-se que estes números pudessem corresponder a um código ASCII, por isso utilizou-se outra fez a tabela ASCII para fazer a conversão, mas desta vez converteu-se a partir de **decimal** e não **hexadecimal**, obtendo a flag:

**picoCTF{g00d\_k1tty!\_n1c3\_k1tty!\_3d84edc8}**