15 minutes

IPv4

O protocolo IPv4 possui 32 bits de largura (de 0 a 31). Como cada byte tem 8 bits, podemos dizer que o cabeçalho IPv4 possui 4 bytes de largura, e cada espaço existente em cada linha no cabeçalho deverá receber um algarismo binário.

Análise de Tráfego em Redes TCP/IP

Cabeçalho IPv4

```
| - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -|
|    1 byte      |    2 byte     |   3 byte      |
4 byte      |
| 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0
1 2 3 4 5 6 7|
| - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -|
|version |  IHL  |Type of service|       Total
Lenght            |
| - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -|
|      Identification           |Flag | Fragment
Offset         |
| - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -|
| Time to Live   | Protocol      |    Header
Checksum           |
| - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -|
```

- **Campo Version**

  O campo version possui largura de 4 bits e é responsável por informar qual é a versão do IP, sendo assim no IPv4 sempre teremos o conteúdo binário `0100` que representa 4 em números binários.

- **Campo IHL (Internet Header Lenght)**

  Este campo pode ser traduzido para *tamanho do cabeçalho internet*, também possui 4 bits e é responsável por informar quantas linhas há no cabeçalho. Todos os elementos das 5 primeiras linhas do cabeçalho IPv4 são de existência obrigatórios, isso significa dizer que o mínimo de linhas de um cabeçalho IP será cinco. Em outras palavras o cabeçalho IP terá no mínimo `20` bytes (5 linhas x 5 bytes por linhas). Em resumo o campo IHL deve ter no mínimo `20` bytes e no máximo `60` bytes, essa variação se dará devido ao campo `options` .

- **Campo Type of Service**

O campo `ToS` , que significa tipo de serviço, possui 8 bits (1 byte) de tamanho e é responsável por determinar algumas características para o tráfego de dados. com vista a melhorar a qualidade do serviço (`QoS`). Este campo basicamente indica para os roteadores que estarão no caminho do pacote como tal deverá ser tratado, se o roteador suportar `ToS` ele obedererá às indicações. É fato que a maioria dos roteadores ignora o conteúdo deste campo, pois o `QoS` é comumente tratado por outros protocolos, este campo não é relevante para a análise de tráfego.

- **Campo Total Lenght** É o campo que contém o tamanho total do pacote IP (Cabeçalho + Payload), pode ser formado por 16 bits, teoricamente, admite valores de 0 a 65535 em decimal, porém deve ser subtraído desse valor o cabeçalho IP que não pode ser menor que 20 bytes e o payload que deve ter no mínimo 1 byte de dados, então o menor valor para esse campo será 21 em decimal. Em resumo, um pacote IP poderá ter até 65535 bytes de tamanho, isso equivale a, aproximadamente 64KB.

- **Campo Identification** Este é o primeiro campo da segunda linha do cabeçalho IPv4, a qual trata da fragmentação de pacotes. Este campo identifica os pacotes IP, cada pacote deverá ter um número de identificação diferente que vai de 0 a 65535 em decimal e seu valor é criado a cada nova conexão, com base em algoritmos no kernel do SO, dentro da mesma conexão os pacotes terão números seriais. Apenas para entender o porque da fragmentação, as redes mais comuns são do tipo ethernet, que, normalmente só conseguem transmitir 1500 bytes de cada vez, Caso o IP a ser transmitido tenha, por exemplo, 3500 bytes será necessário quebrá-lo em pedaços.

- **Campo Flag:** Como dito anteriormente a segunda linha do cabeçalho IPv4 é destinada a controlar a fragmengação, o campo `flags` é responsável por ajudar a controlar o fluxo de fragmentos IP, quando ocorre a fragmentação. Quando fragmentados, os pacotes possuem o mesmo cabeçalho e por consequencia o mesmo número de identificação (`Identification`), daí surge a próxima medida de controle: as `flags`. Flag é um campo que possui 3 bits, e são eles:

0 1 2

none DF MF

- `none` : esse bit é reservado para uso futuro e, por enquanto, sempre terá valor zero.
- `DF` : Significa Don’t Fragment. Esse bit é utilizado para expressar se poderá ou não haver fragmentação do pacote IP. Caso o bit seja marcado com valor 1, o pacote não poderá sofrer fragmentação por roteadores que estejam no caminho. Assim, se o pacote for maior do que capacidade de um determinado roteador, este enviará uma solicitação de retransmissão em partes menores. Tal solicitação será feita pelo protocolo ICMP. Caso o bit `DF` esteja marcado como 0,

os roteadores existentes no caminho poderão realizar a quebra do pacote, então os valores possíveis são `0` (pode fragmentar) e 1 (não pode fragmentar).

- `MF` : Significa More Fragment. Caso seja necessário fragmentar um pacote, o bit MF indicará, no destino, a cada fragmento recebido, se haverá mais fragmentos que chegarão ou não. O bit 1 representa mais fragmentos. Exemplo: ao dividir um pacote em 3 partes, as duas primeiras terão o bit MF marcado com 1, enquanto a última será marcada com MF `0`.

> A flag `MF` só poderá receber o bit 1 se `DF` for igual a `0`.

- **Campo Fragment Offset:** É o último elemento de controle na fragmentação. O Fragment Offset é um campo de 13 bits que se refere ao byte inicial de cada fragmento do pacote dividido por 8. O Cálculo sempre irá considerar apenas o payload. * Como exemplo, vamos manipular um pacote de `3000` bytes. Se cada fragmento estiver limitado a `1500` bytes de tamanho total e o cabeçalho tiver `20` bytes, o payload só poderá ter, no máximo, `1480` bytes. Portanto, o payload do primeiro fragmento irá de `0` a `1479`. Então considerando o primeiro byte do payload do primeiro fragmento, que estará na posição `0`, teremos `Fragment Offset` será `0` / 8= `0`. A segunda parte dp payload irá de `1480` a `2959`. Então `Fragment Offset` será `1480` /8 = `185`. O terceiro e último pedaço irá de `2960` a `2979`, ou seja, `Fragment Offset` será `2969` / 8 = `370`. Com isso, será possível remontar o pacote na ordem certa, bastando colocar os fragmentos em ordem crescente relativa ao `Fragment Offset` , a ordem final de montagem será `0`, `185`, `375`.

- **Análise de uma fragmentação IP** Vamos realizar uma análise com auxílio do *tcpdump*. Será enviado um pacote gerado com o comando ping, mostrado a seguir.

```
keilon@ubuntu:~$ ping -c 1 -s 3500 192.168.1
```

Este comando gerou um pacote com `3500` bytes de payload. Em uma rede ethernet com taxa de transmissão de `1500` bytes, esse pacote terá que ser fragmentado em três pedaços. Para fazer a anãlise vamos analisar a saída do *tcpdump*:

```
keilon@ubuntu:~$ sudo tcpdump -n icmp and ho
tcpdump: listening on enp6s0, link-type EN10

❶  00:41:17.773004 IP (tos 0x0, ttl 64, id 3
       192.168.1.10 > 192.168.1.1: ICMP echo

❷  00:41:17.773016 IP (tos 0x0, ttl 64, id 3
       192.168.1.10 > 192.168.1.1: ip-proto-

❸  00:41:17.773018 IP (tos 0x0, ttl 64, id 3
       192.168.1.10 > 192.168.1.1: ip-proto-

❹  00:41:17.773986 IP (tos 0x0, ttl 64, id 2
       192.168.1.1 > 192.168.1.10: ICMP echo

❺  00:41:17.774106 IP (tos 0x0, ttl 64, id 2
       192.168.1.1 > 192.168.1.10: ip-proto-

❻  00:41:17.774141 IP (tos 0x0, ttl 64, id 2
       192.168.1.1 > 192.168.1.10: ip-proto-
```

Temos duas linhas por pacote, a primeira se refere ao IP e a segunda ao ICMP. As linhas de 1 a 3 estão mostrando o envio do ping, conhecido como *ICMP echo request*, enquanto 4 a 6 mostram a resposta a este, ou seja um *ICMP echo replay*. Na linha 1, vemos que a máquina 192.168.1.10 enviando um ping para 192.168.1.1, o campo identification tem como valor 3581 e *Fragment Offset* é 0. > O *tcpdump* mostra o valor de *Fragment Offset* multiplicado por 8, com isso não vemos o *offset* puro, e sim o byte real que o gerou.

Ainda na linha 1, *flags [+]* represente que MF é igual a 1. Então, teremos mais fragmentos depois deste, isto é a prova de que o pacote foi fragmentado. O tamanho total o do fragmento atual (cabeçalho + payload) é igual a 1500 bytes. O tamanho do payload, que é o protocolo ICMP, é 1480 bytes. Os ICMP echo replay e echo request tem seus próprios campos de identificação. A linha 2 mostra que o identification do IP é 3581, já sabemos que todos os fragmentos de um pacote IP possuem o mesmo número de identificação, o mesmo campo *Protocol*, que, no caso corresponde ao ICMP, mesmo endereço de origem e destino. Se observarmos todos os campos são iguais nas linhas 1, 2 e 3. Isso

confirma que temos três fragmentos de um mesmo pacote IP. Ainda na linha 2 é possível ver o *Fragment Offset* 1480 e a flag *MF* marcada com 1 novamente, na linha 3 o *offset* é 2960 e *MF* está desabilitado, indicando que não há mais fragmentos a serem recebidos. As linhas de 4 a 6 mostram a responta do ping.

- **Campo Time to Live (TTL)** Este campo se refere ao tempo de vida de um pacote, inicialmente os pacotes tem um valor de TTL pré-determinado geralmente pelo kernel do SO, a cada roteador (HOP) que o pacote passa é decrementado o valor 1 do TTL que quando chega a 0 é descartado pelo roteador, evitando deste modo que o pacote circulasse eternamente na rede de computadores.

O comando que mostra explicitamente o *TTL* é o ping, que serve para tester a conectividade entre hosts.

```
keilon@ubuntu:~$ ping -c 4 www.google.com
PING www.google.com (172.217.30.164) 56(84) bytes of
64 bytes from 172.217.30.164: icmp_seq=1 ttl=115 tim
64 bytes from 172.217.30.164: icmp_seq=2 ttl=115 tim
64 bytes from 172.217.30.164: icmp_seq=3 ttl=115 tim
64 bytes from 172.217.30.164: icmp_seq=4 ttl=115 tim

--- www.google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, t
rtt min/avg/max/mdev = 19.471/19.812/19.959/0.199 ms
```

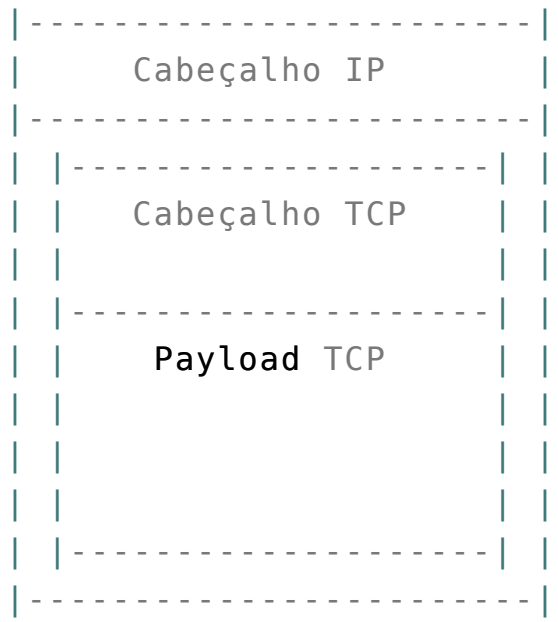
- O resultado do comando acima é a respota do Google, ou seja, o echo reply. Muito provavelmente este servidor web tem o TTL 128 e foi decrementando até chegar em 115, passando por 13 roteadores. O campo TTL possui 8 bits e pode ir de 0 a 25, cada sistema operacional tem um TTL padrão que pode ser alterado.

```
keilon@ubuntu:~$ traceroute www.google.com
traceroute to www.google.com (172.217.172.132), 64 h
  1   192.168.1.1  0,397ms  0,326ms  0,351ms
  2   177.66.167.170  1,340ms  0,899ms  0,907ms
  3   172.16.167.5  1,274ms  1,745ms  1,052ms
  4   189.125.215.253  2,902ms  1,878ms  2,151ms
```

```
 5   64.209.11.186  18,873ms  18,946ms  19,048ms
 6   72.14.212.213  21,351ms  22,012ms  22,109ms
 7   *  *  *
 8   209.85.242.198  19,827ms  20,025ms  19,926ms
 9   172.253.66.25  22,137ms  21,859ms  21,961ms
10   74.125.243.65  21,093ms  20,964ms  20,779ms
11   172.253.66.23  20,192ms  19,913ms  19,960ms
12   172.217.172.132  19,810ms  19,969ms  19,916ms
```

Acima um exemplo usando o traceroute, que mostra a rota do pacote até o host de destino. Quando uma linha mostrar mais de um roteador, significa que temos roteadores trabalhando em paralelo fazendo balanceamento de carga.

- **Campo Protocol** Este campo é responsável por dizer qual será o conteúdo encontrado no payload do IP. No payload do IP só poderá existir um tipo de elemento: outro protocolo, em outras palavras o IP sempre irá carregar outro protocolo no seu payload, os protocolos que podem ser encapsulados pelo IP são conhecidos como `protocolos IP` .



A figura acima representa o funcionamento do protocolo IP, sempre encontraremos dentro dele, outro protocolo como payload. O campo `Protocol` possui 8 bits, e por isso admite valores de `0` a 255, o protocolo ICMP é o número 1, enquanto o TCP é o número 7 e o UDP o número 17.

- **Campo Header Checksum**

O campo `header checksum` é responsável por garantir que o pacote IP transite integro. Esse campo pode nos dizer por exemplo, se durante o tráfego algum equipamento truncou bits. O `Header Checksum` contém 16 bits e vai de `0` a 65535 e é calculado com base nos campos existentes em todo cabeçalho IP, em resumo é feito um cálculo de todos os campos do cabeçalho IP e este valor fica registrado dentro do campo header checksum.

- **Campo Source Address** O `Source Address` é o campo que se refere ao endereço do host que está enviando o pacote IP, o campo contém 32 bits de largura, ou seja 4 bytes. Os campos `Source Address` e `Destination Address` são os maiores campos do cabeçalho IP, e o motivo pelo qual foi escolhido 32 bits de largura para o cabeçalho.

- **Campo Destination Address**

  É similar ao Source Address, porém contém o endereço de destino e é o último campo obrigatório do cabeçalho IP, tendo até aqui `20` bytes de cabeçalho.

- **Campo Options**

  Este campo não é de existência obrigatória no pacote IP, seu tamanho irá variar de acordo com a quantidade de informações que ele terá, podendo ir de `0` a `40` bytes, o que corresponde de `0` a `10` linhas de cabeçalho.

  Este campo não é tão importante para a análise de tráfego, ele só é relevante em análises muito avançadas. Apenas por curiosidade este campo pode trazer as seguintes informações: - Informações de segurança e restrições de acesso e manipulação do pacote; - Relação de roteadores transpostos para chegar a um destino `(Record Route)` ; - data e hora `(time stamp)` - Loose source routing - Strict source routing

- **Campo Padding**

  Este campo funciona como uma “rolha”, e só existirá caso o campo options também exista, a função dele é muito simples, como o campo options tem tamanho variados, seu término geralmente não irá coincidir com o fim da linha do cabeçalho. Em outras palavras, cada linha do cabeçalho deve ter

obrigatoriamente 4 bytes, assim o `padding` entra para completar, caso necessário, a linha options.

- **Payload**

  O campo payload de um pacote IP nada mais é do que a área de dados. É nessa área que serão colocados os dados que trafegarão dentro dele na rede. O tamanho do payload irá variar de acordo com a quantidade de informações que nele será colocada. É importante ressaltar que o menor cabeçalho IP contém 20 bytes de tamanho e que o maior IP possível incluindo o cabeaçalho tem 65535 bytes. Assim o maior payload IP possível terá 65535 - 20 = 65515 bytes. Considerando que um pacote Ip deverá conter outro protocolo dentro de seu payload, este nunca estará vazio. Com isso, podemos dizer que o menor tamanho possível para um payload IP é 1 byte.

  > Onde está a máscara de rede e o CIDR? Ambos fazem parte de procedimentos internos e não trafegam na rede. Note que eles nem aparecem nos campos do protocolo IP.

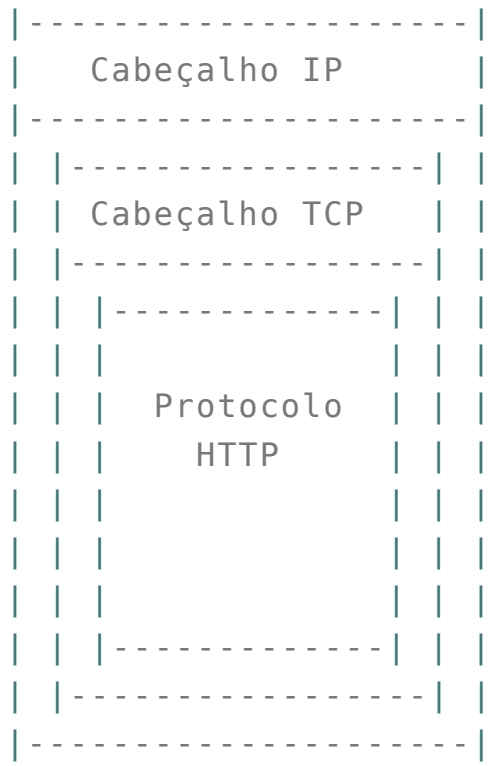
- **Características do IPv4**

  - `Não garantia de entrega` - O protocolo IP é do tipo *best effort*, ou seja,não garante, por si só, que os dados serão entregues ao destinatário. Para o caso de garantia de entrega é necessário outro protocolo, o TCP por exemplo.
  - `Não garantia de payload` - A única verificação que o protocolo IP faz é a do cabeçalho, por intermédio do *header checksum*, então o IP não é capaz de garantir os dados dentro de seu payload, outros protocolos farão essa garantia.
  - `Endereço Organizado` - Trata-se do endereçamento IP, possui um método extremamente flexível e organizado.
  - `Fragmentação de Pacote` - Os pacotes podem ser fragmentados sempre que necessário, o que facilita seu tráfego por diversos tipos de topologias, há um mecanismo de controle para remontagem.
  - `QoS não garantido` - O IP pode utilizar o “Quality of service”, provido pelo campo `Type of Service` ou `Differentiated Services` , com isso é possível priorizar tráfego e atribuir graus de importância aos pacotes, todavia,

geralmente os roteadores ignoram esse campo, com isso não é possível garantir que haja QoS.
    - `Tempo de Vida` - Esta característica é provida pelo campo TTL, é essencial para que um pacote não trafegue eternamente pelas redes.
    - `Container de protocolos` - O IP, carrega dentro de si outros protocolos.
    - `Roteamento` - O pacote IP contém as informações necessárias para chegar a seu destino, campo Destinarion Address já é suficiente para isso.

- **Considerações gerais sobre o IPv4**

  Cada protocolo IP tem uma importância e executa alguma função especial: transportar outros protocolos. São eles o TCP e UDP. Em outras palavras, o TCP e o UDP recebem outros protocolos no seu payload, como o HTTP.



- **Conclusão** O protocolo IP funciona como um grande caminhão, a boléia representa o cabeçalho, onde está todo o controle de tráfego. A corroceria é análoga ao payload, onde estão os dados que serão carregados. O protocolo IP sempre terá outro protocolo em seu payload. A todos os protocolos que podem ser transportados pelo IP damos o nome de `Protocolos IP` .

#IPv4

3175 Words

2020-06-14 05:36

← PCF to VPNC