# HTTP

- Comece a capturar pacotes com o wireshark
- Acesse o seguinte link: http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html
- Dê um refresh(f5) na página
- Pare a captura
- Aponte as diferenças nas duas respostas HTTP obtidas e explique o que aconteceu

R:

Wireshark:

```
 382 2.390112    192.168.0.11      128.119.245.12    HTTP    540 GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1
 502 2.594498    128.119.245.12    192.168.0.11      HTTP    784 HTTP/1.1 200 OK  (text/html)
1591 5.138900    192.168.0.11      128.119.245.12    HTTP    652 GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1
1706 5.346992    128.119.245.12    192.168.0.11      HTTP    293 HTTP/1.1 304 Not Modified
```

Primeiro contato:

Na primeira requisição recebemos o response, normalmente.

Segundo contato (refresh):

Código de resposta 304 (not modified)

O código de resposta HTTP de redirecionamento do cliente **304 Not Modified** indica que não há necessidade de retransmitir a requisição de recursos. É um redirecionamento implícito para o recurso em cache.

A partir do refresh, enviamos ao servidor ao um pacote que solicita o conteúdo, mas enviamos também uma informação que diz “se o contato não foi alterado não precisa reenviar”, neste caso reutilizamos o conteúdo salvo em cache.

```
Hypertext Transfer Protocol
  GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1\r\n
  Host: gaia.cs.umass.edu\r\n
  Connection: keep-alive\r\n
  Cache-Control: max-age=0\r\n
  Upgrade-Insecure-Requests: 1\r\n
  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/100.0.4896.127 Safari/537.36\r\n
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9\r\n
  Accept-Encoding: gzip, deflate\r\n
  Accept-Language: en-US,en;q=0.9,pt;q=0.8\r\n
  If-None-Match: "173-5dcbf3948748f"\r\n
  If-Modified-Since: Sat, 16 Apr 2022 05:59:01 GMT\r\n
  \r\n
  [Full request URI: http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html]
  [HTTP request 2/2]
  [Prev request in frame: 382]
```

# HTTP

- Comece a capturar pacotes com o wireshark
- Acesse o seguinte link: http://gaia.cs.umass.edu/wireshark-labs/protected_pages/HTTP-wireshark-file5.html
- Insira os dados pedidos:
  - username:wireshark-students
  - password:network
- Pare a captura
- Analise as capturas e explique como os dados foram passados pelo protocolo

Wireshark:

| | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 63 | 7.488138 | 192.168.0.11 | 128.119.245.12 | HTTP | 546 | GET /wireshark-labs/protected_pages/HTTP-wireshark- HTTP/1.1 |
| 71 | 7.682004 | 128.119.245.12 | 192.168.0.11 | HTTP | 771 | HTTP/1.1 401 Unauthorized (text/html) |
| 152 | 11.220518 | 192.168.0.11 | 128.119.245.12 | HTTP | 631 | GET /wireshark-labs/protected_pages/HTTP-wireshark- HTTP/1.1 |
| 159 | 11.416367 | 128.119.245.12 | 192.168.0.11 | HTTP | 573 | HTTP/1.1 404 Not Found (text/html) |

O protocolo http não possui criptografia, no site fizemos login com “wireshark-students” e senha “network”, ou seja, enviamos nossas credenciais para o servidor sem criptografia. Com o uso do wireshark, capturamos os pacotes enviados para o servidor e conseguimos ter acesso as credenciais enviadas, vale destacar que o “Basic d2lyZXNoYXJrLXN0dWRlbnRzOm5ldHdvcms=” diz respeito a uma string encodada em basex64, não é criptografia.

```
Destination Port: 80
[Stream index: 2]
[Conversation completeness: Incomplete, DATA (15)]
[TCP Segment Len: 577]
Sequence Number: 493    (relative sequence number)
Sequence Number (raw): 2961905161
[Next Sequence Number: 1070    (relative sequence number)]
Acknowledgment Number: 718    (relative ack number)
Acknowledgment number (raw): 3682917275
0101 .... = Header Length: 20 bytes (5)
> Flags: 0x018 (PSH, ACK)
Window: 1023
[Calculated window size: 261888]
[Window size scaling factor: 256]
Checksum: 0x3893 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
> [Timestamps]
> [SEQ/ACK analysis]
TCP payload (577 bytes)
Hypertext Transfer Protocol
> GET /wireshark-labs/protected_pages/HTTP-wireshark- HTTP/1.1\r\n
Host: gaia.cs.umass.edu\r\n
Connection: keep-alive\r\n
Cache-Control: max-age=0\r\n
v Authorization: Basic d2lyZXNoYXJrLXN0dWRlbnRzOm5ldHdvcms=\r\n
    Credentials: wireshark-students:network
Upgrade-Insecure-Requests: 1\r\n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/100.0.4896.127 Safari/537.36\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9\r\n
Accept-Encoding: gzip, deflate\r\n
Accept-Language: en-US,en;q=0.9,pt;q=0.8\r\n
\r\n
[Full request URI: http://gaia.cs.umass.edu/wireshark-labs/protected_pages/HTTP-wireshark-]
[HTTP request 2/2]
[Prev request in frame: 63]
[Response in frame: 159]
```

.

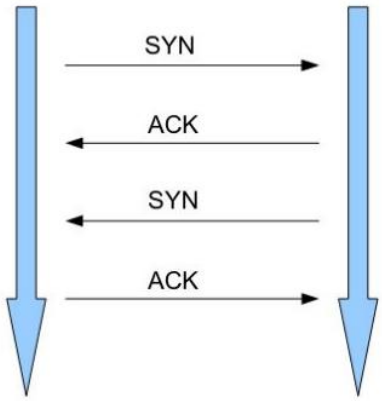
# TCP

- Faça a análise de alguma das capturas anteriores e identifique os pacotes correspondentes ao Three-way-handshake

No protocolo TCP, ocorre o three-way-handshake:

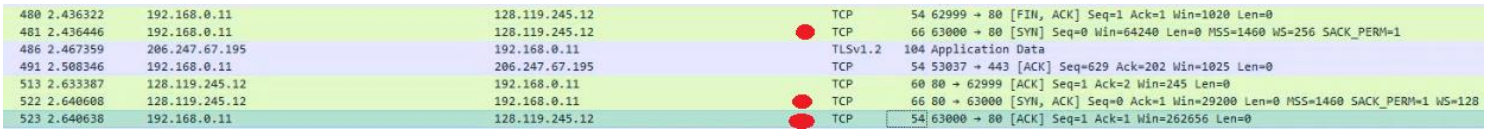
1) Enviamos o SYN
2) Recebemos o ACK e SYN
3) Enviamos o ACK

Com isso, temos a conexão estabelecida com sucesso




No wireshark:

com o destaque em vermelho, notamos o as três etapas destacadas anteriormente, 1) enviamos o SYN; 2) recebemos ACK e SYN; 3) enviamos o ACK




em “2) ACK e SYN”, podemos ver que esta sendo settada a comunicação, por meio da sequencia e o tamanho da janela. Vale destacar que o protocolo TCP admite mudanças devido ao seu comportamento “serrilhado”, ou seja, o tamanho da janela é dinâmico, podendo sofrer alteração ao longo do envio dos pacotes.

# DNS

- Comece a capturar pacotes com o wireshark
- Utilize algum comando para fazer consultas DNS (dig, nslookup, host ...) e faça duas consultas a algum site de sua escolha. Uma requisição do tipo **A** e outra do tipo **NS**
- Pare a captura
- Faça a análise das respostas obtidas e o seu significado

O protocolo DNS serve para traduzir o nome do site, o url, em endereços de ip.

No google.com:

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 15 | 2.834765 | 2804:14d:5cd3:a828:a1fb:7743:cfda:ce9c | 2804:14d:1:0:181:213:132:3 | DNS | 110 | Standard query 0xabb1 A smartscreen-prod.microsoft.com |
| 16 | 2.844775 | 2804:14d:1:0:181:213:132:3 | 2804:14d:5cd3:a828:a1fb:7743:cfda:ce9c | DNS | 498 | Standard query response 0xabb1 A smartscreen-prod.microsoft.com CNAME wd-prod-ss.trafficmanager.net CNAME wd-prod-ss-br-south-1-fe.brazilsou |
| 30 | 2.923902 | 2804:14d:5cd3:a828:a1fb:7743:cfda:ce9c | 2804:14d:1:0:181:213:132:3 | DNS | 109 | Standard query 0xc39b A v10.events.data.microsoft.com |
| 31 | 2.923995 | 2804:14d:5cd3:a828:a1fb:7743:cfda:ce9c | 2804:14d:1:0:181:213:132:3 | DNS | 109 | Standard query 0x0065 AAAA v10.events.data.microsoft.com |
| 32 | 2.932340 | 2804:14d:1:0:181:213:132:3 | 2804:14d:5cd3:a828:a1fb:7743:cfda:ce9c | DNS | 550 | Standard query response 0xc39b A v10.events.data.microsoft.com CNAME global.asimov.events.data.trafficmanager.net CNAME onedscolprdcus00.cen |
| 33 | 2.933251 | 2804:14d:1:0:181:213:132:3 | 2804:14d:5cd3:a828:a1fb:7743:cfda:ce9c | DNS | 280 | Standard query response 0x0065 AAAA v10.events.data.microsoft.com CNAME global.asimov.events.data.trafficmanager.net CNAME onedscolprdwus03. |
| 84 | 3.584942 | 2804:14d:5cd3:a828:a1fb:7743:cfda:ce9c | 2804:14d:1:0:181:213:132:3 | DNS | 90 | Standard query 0x0005 NS google.com |
| 85 | 3.596520 | 2804:14d:1:0:181:213:132:3 | 2804:14d:5cd3:a828:a1fb:7743:cfda:ce9c | DNS | 338 | Standard query response 0x0005 NS google.com NS ns2.google.com NS ns4.google.com NS ns1.google.com NS ns3.google.com A 216.239.32.10 A 216.2 |
| 96 | 4.037402 | 2804:14d:5cd3:a828:a1fb:7743:cfda:ce9c | 2804:14d:1:0:181:213:132:3 | DNS | 117 | Standard query 0xbd3b A array810.prod.do.dsp.mp.microsoft.com |
| 97 | 4.037499 | 2804:14d:5cd3:a828:a1fb:7743:cfda:ce9c | 2804:14d:1:0:181:213:132:3 | DNS | 117 | Standard query 0xbf77 AAAA array810.prod.do.dsp.mp.microsoft.com |
| 98 | 4.045689 | 2804:14d:1:0:181:213:132:3 | 2804:14d:5cd3:a828:a1fb:7743:cfda:ce9c | DNS | 399 | Standard query response 0xbd3b A array810.prod.do.dsp.mp.microsoft.com A 20.190.9.86 NS ns1-06.azure-dns.com NS ns4-06.azure-dns.info NS ns3 |
| 99 | 4.046819 | 2804:14d:1:0:181:213:132:3 | 2804:14d:5cd3:a828:a1fb:7743:cfda:ce9c | DNS | 190 | Standard query response 0xbf77 AAAA array810.prod.do.dsp.mp.microsoft.com SOA ns1-06.azure-dns.com |

No youtube.com:

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 59 | 10.270388 | 2804:14d:5cd3:a828:a1fb:7743:cfda:ce9c | 2804:14d:1:0:181:213:132:3 | DNS | 91 | Standard query 0x0006 NS youtube.com |
| 60 | 10.280870 | 2804:14d:1:0:181:213:132:3 | 2804:14d:5cd3:a828:a1fb:7743:cfda:ce9c | DNS | 346 | Standard query response 0x0006 NS youtube.com NS ns4.google.com NS ns3.google.com NS ns1.google.com NS ns2.google.com AAAA 2001:4860:4802:32 |
| 69 | 12.118940 | 2804:14d:5cd3:a828:a1fb:7743:cfda:ce9c | 2804:14d:1:0:181:213:132:3 | DNS | 95 | Standard query 0x56f4 A www.youtube.com |
| 70 | 12.127932 | 2804:14d:1:0:181:213:132:3 | 2804:14d:5cd3:a828:a1fb:7743:cfda:ce9c | DNS | 289 | Standard query response 0x56f4 A www.youtube.com CNAME youtube-ui.l.google.com A 142.250.218.110 A 142.250.218.142 A 142.251.132.78 A 142.25 |

No twitter.com:

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 7 | 2.694069 | 2804:14d:5cd3:a828:a1fb:7743:cfda:ce9c | 2804:14d:1:0:181:213:132:3 | DNS | 91 | Standard query 0x0007 NS twitter.com |
| 8 | 2.710765 | 2804:14d:1:0:181:213:132:3 | 2804:14d:5cd3:a828:a1fb:7743:cfda:ce9c | DNS | 458 | Standard query response 0x0007 NS twitter.com NS d01-02.ns.twtrdns.net NS a.r06.twtrdns.net NS c.r06.twtrdns.net NS d.r06.twtrdns.net NS ns4 |

```
    Additional RRs: 7
  ▾ Queries
    ▸ twitter.com: type NS, class IN
  ▾ Answers
    ▸ twitter.com: type NS, class IN, ns d01-02.ns.twtrdns.net
    ▸ twitter.com: type NS, class IN, ns a.r06.twtrdns.net
    ▸ twitter.com: type NS, class IN, ns c.r06.twtrdns.net
    ▸ twitter.com: type NS, class IN, ns d.r06.twtrdns.net
    ▸ twitter.com: type NS, class IN, ns ns4.p34.dynect.net
    ▸ twitter.com: type NS, class IN, ns ns1.p34.dynect.net
    ▸ twitter.com: type NS, class IN, ns ns2.p34.dynect.net
    ▸ twitter.com: type NS, class IN, ns d01-01.ns.twtrdns.net
    ▸ twitter.com: type NS, class IN, ns b.r06.twtrdns.net
    ▸ twitter.com: type NS, class IN, ns ns3.p34.dynect.net
  ▸ Additional records
```

Vimos que, nos 3 casos, o destino foi o mesmo, isso se deve ao fato de que, para otimizar o tempo de resposta, as próprias operadores tem uma lista de dns frequentemente acessados, para que não seja necessário consultar um servidor distante a cada conexão feita.

- Suponha que você recebeu a faixa de endereços 146.164.70.0/23. A quantas máquinas você pode atribuir endereços IP públicos, com esta faixa? Quantas sub-redes IP você pode criar, no máximo? Neste caso, quantas máquinas no máximo poderiam receber endereços IP públicos? Justifique as respostas.

Ip: 146.164.70.0 mascara:/23

Mascara:

11111111 . 11111111 . 1111111 0 . 00000000

Para acharmos o endereço que serve para identificar a rede, fazemos um and (bit a bit) da mascara com o ip.

Rede = Resultado do and bit a bit: 10010010 . 10100100 . 1000110 0 . 00000000

Decimal 146 . 164 . 70 . 0

Endereços de IPS públicos = ($2^9 - 2$), devido ao endereço de rede e ao broadcast que não podem ser atribuídos a algum host.

Range: 146 . 164 . 70 . 0 até

146 . 164 . 71 . 255

Analisando a mudança da mascara /23 para a mascara /24:

- ganhamos 2 sub-redes para cada sub-rede

11111111 . 11111111 . 11111111 . 00000000

mascara

| | | | | | | |
|---|---|---|---|---|---|---|
| sub-rede1: | 146 | . | 164 | . | 70 | . 0 |
| | 146 | . | 164 | . | 70 | . 255 |
| Sub-rede2: | | | | | | |
| | 146 | . | 164 | . | 71 | . 0 |
| | 146 | . | 164 | . | 71 | . 255 |

Analisando a mudança da mascara /24 para a mascara /25:

- ganhamos 2 sub-redes para cada sub-rede

| | | | | | | |
|---|---|---|---|---|---|---|
| sub-rede1: | 146 | . | 164 | . | 70 | . 0 |
| | 146 | . | 164 | . | 70 | . 127 |
| Sub-rede2: | | | | | | |
| | 146 | . | 164 | . | 70 | . 128 |
| | 146 | . | 164 | . | 70 | . 255 |
| sub-rede3: | 146 | . | 164 | . | 71 | . 0 |
| | 146 | . | 164 | . | 71 | . 127 |
| Sub-rede4: | | | | | | |
| | 146 | . | 164 | . | 71 | . 128 |
| | 146 | . | 164 | . | 71 | . 255 |

Seguindo a lógica anterior, a maior quantidade de sub-redes, pode ser dada por:

Ip: 146.164.70.0 mascara 30

Mascara:

11111111 . 11111111 . 11111111 . 11111100

146 . 164 . 70 . 0 até

146 . 164 . 71 . 255

Se 11111111 . 11111111 . 11111110 . 00000000 → rede1

Se 11111111 . 11111111 . 11111110 . 00000100 → rede2

Se 11111111 . 11111111 . 11111110 . 00001000 → rede3

...

Se 11111111 . 11111111 . 11111111 . 11111100 → rede$2^7$

Perdemos 2 endereços a cada nova sub-rede, referente ao broadcast e rede. Com isso, se possuímos $2^7$ sub-redes, possuímos $2*2^7$ não posso usar, sendo que cada sub-rede armazena até 2 hosts, seguindo a mesma lógica, possuímos $2*2^7$ hosts com ip público.