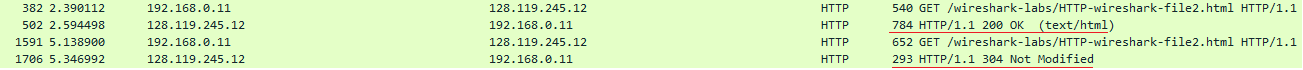


R:

Wireshark:

Primeiro contato:

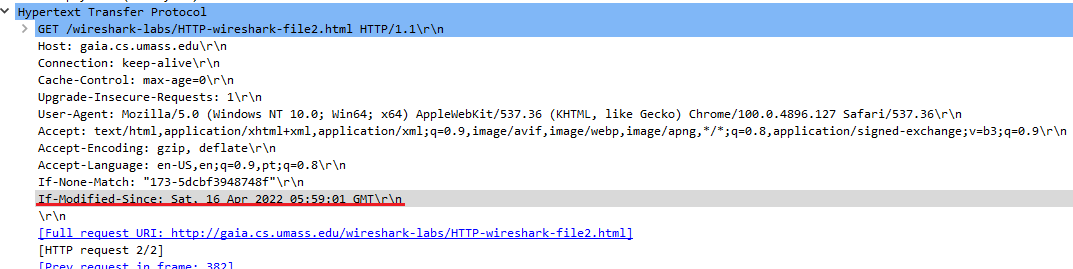
Na primeira requisição recebemos o response, normalmente.

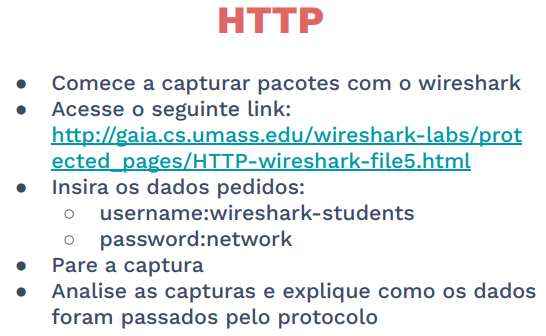
Segundo contato (refresh):

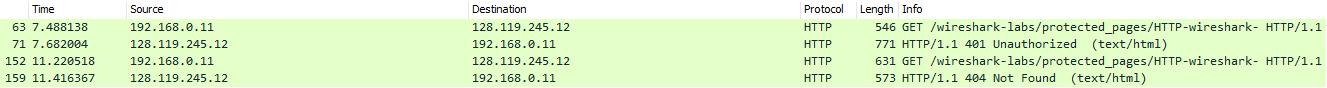
Código de resposta 304 (not modified)

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.

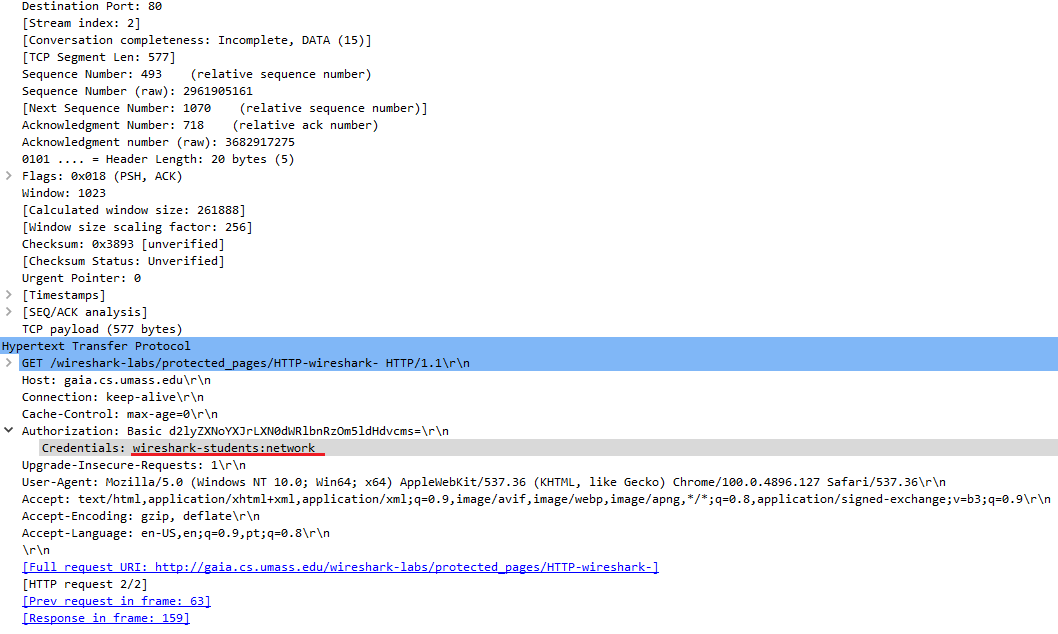
A partir do refresh, enviamos ao servidor ao um pacote que solicita o conteúdo, mas enviamos também uma informação que diz “se o contato não foi alterado não precisa reenviar”, neste caso reutilizamos o conteúdo salvo em cache.

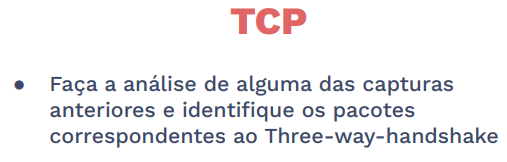




Wireshark:

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.

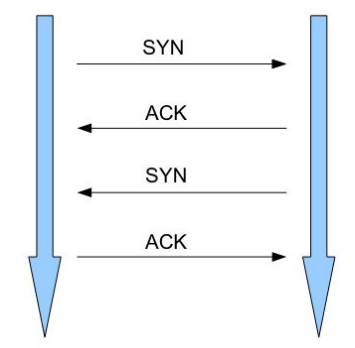
.



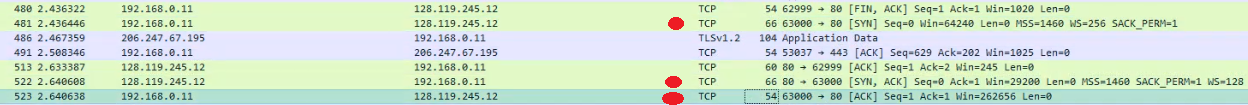
No protocolo TCP, ocorre o three-way-handshake:

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

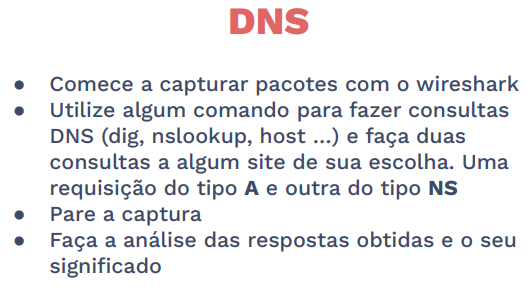
Com isso, temos a conexão estabelecida com sucesso



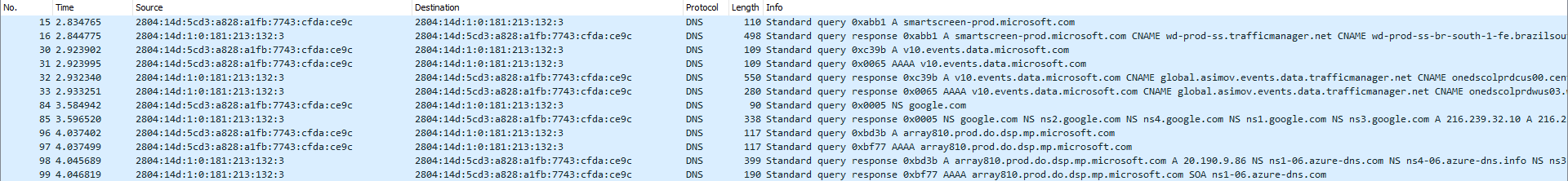
No wireshark:

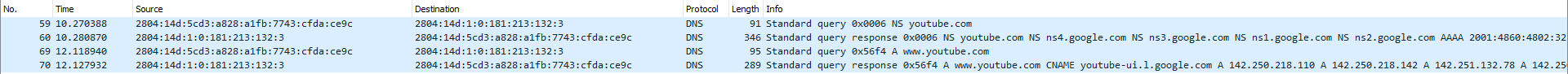
com o destaque em vermelho, notamos o as três etapas destacadas anteriormente, 1) enviamos o SYN; 2) recebemos ACK e SYN; 3) enviamos o ACK

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.

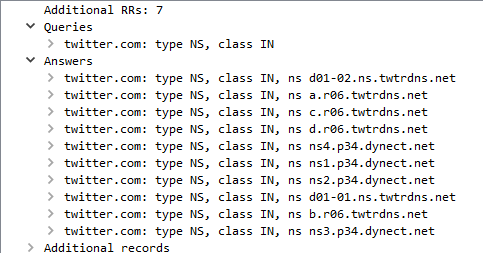


O protocolo DNS serve para traduzir o nome do site, o url, em endereços de ip.

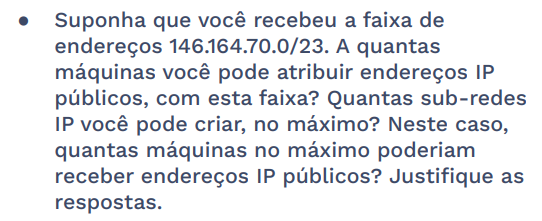
No google.com:

No youtube.com:

No twitter.com:



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.



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 = (29 – 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 🡪 rede27

Perdemos 2 endereços a cada nova sub-rede, referente ao broadcast e rede. Com isso, se possuímos 27 sub-redes, possuímos 2\*27 não posso usar, sendo que cada sub-rede armazena até 2 hosts, seguindo a mesma lógica, possuímos 2\*27 hosts com ip público.