1.Introdução

HTTP é o que é usado sempre que você visualiza um site, desenvolvido por Tim Berners-Lee e sua equipe entre 1989-1991. HTTP é o conjunto de regras utilizadas para comunicação com servidores web para transmissão de dados de páginas web, sejam HTML, imagens, vídeos, etc.

HTTPS é a versão segura do HTTP. Os dados HTTPS são criptografados, portanto, não apenas impedem que as pessoas vejam os dados que você está recebendo e enviando, mas também oferecem garantias de que você está se comunicando com o servidor da Web correto e não com algo que se faz passar por ele.

Quando acessamos um site, seu navegador precisará fazer solicitações a um servidor web para ativos como HTML, imagens e baixar as respostas. Antes disso, você precisa informar ao navegador especificamente como e onde acessar esses recursos, é aqui que os URLs vão ajudar.

Uma URL é predominantemente uma instrução sobre como acessar um recurso na Internet e é separada nas seguintes partes:

Scheme	Isso instrui sobre qual protocolo usar para acessar o recurso, como HTTP, HTTPS, FTP (File Transfer Protocol).
Users	Some services require authentication to log in, you can put a username and password into the URL to log in.
Host	O nome de domínio ou endereço IP do servidor que você deseja acessar.
Port	A porta à qual você se conectará, geralmente 80 para HTTP e 443 para HTTPS, mas pode ser hospedada em qualquer porta entre 1 - 65535.
Path	O nome do arquivo ou local do recurso que você está tentando acessar.
Query String	Pedaços extras de informação que podem ser enviados para o caminho solicitado. Por exemplo, /blog?id=1 informaria ao caminho do blog que você deseja receber o artigo do blog com o id 1.
Fragment	Esta é uma referência a um local na página real solicitada. Isso é comumente usado para páginas com conteúdo longo e pode ter uma determinada parte da página diretamente vinculada a ela, de forma que fique visível para o usuário assim que ele acessar a página.

<scheme>://<user>@<host/domain>:<port>/<path><query_string><fragment>

Exemplo de request HTTP

```
GET / HTTP/1.1
Host: tryhackme.com
User-Agent: Mozilla/5.0 Firefox/87.0
Referer: https://tryhackme.com/
```

Linha 1: Esta solicitação está enviando o método GET (mais sobre isso na tarefa Métodos HTTP), solicitando a página inicial com / e informando ao servidor web que estamos usando o protocolo HTTP versão 1.1.

Linha 2: Dizemos ao servidor web que queremos o site tryhackme.com

Linha 3: Informamos ao servidor web que estamos usando o navegador Firefox versão 87

Linha 4: Estamos informando ao servidor web que a página web que nos encaminhou para esta é https://tryhackme.com

Linha 5: As solicitações HTTP sempre terminam com uma linha em branco para informar ao servidor web que a solicitação foi concluída.

Exemplo de resposta HTTP

```
HTTP/1.1 200 OK
Server: nginx/1.15.8
Date: Fri, 09 Apr 2021 13:34:03 GMT
Content-Type: text/html
Content-Length: 98

<html>
<head>
   <title>TryHackMe</title>
</head>
<body>
   Welcome To TryHackMe.com
</body>
</html>
```

Linha 1: HTTP 1.1 é a versão do protocolo HTTP que o servidor está usando e seguida pelo código de status HTTP, neste caso "200 Ok", que nos informa que a solicitação foi concluída com sucesso.

Linha 2: Isso nos informa o software do servidor web e o número da versão.

Linha 3: A data, hora e fuso horário atuais do servidor web.

Linha 4: O cabeçalho Content-Type informa ao cliente que tipo de informação será enviada, como HTML, imagens, vídeos, pdf, XML.

Linha 5: Content-Length informa ao cliente quanto tempo dura a resposta, desta forma podemos confirmar que não faltam dados.

Linha 6: A resposta HTTP contém uma linha em branco para confirmar o fim da resposta HTTP.

Linha 7-14: A informação que foi solicitada, neste caso a página inicial.

2.Métodos HTTP

Os métodos HTTP são uma forma de o cliente mostrar a ação pretendida ao fazer uma solicitação HTTP.

- **GET request:** Isso é usado para obter informações de um servidor web.
- **POST request:** Isso é usado para enviar dados ao servidor web e potencialmente criar novos registros
- **PUT request:** Isso é usado para enviar dados a um servidor web para atualizar informações
- **DELETE request:** Isso é usado para excluir informações/registros de um servidor web.

Quando um servidor HTTP responde, a primeira linha sempre contém um código de status informando ao cliente o resultado de sua solicitação e também potencialmente como lidar com ela. Esses códigos de status podem ser divididos em 5 intervalos diferentes:

Faixa de código	Descrição da faixa
100 até 199 - Information Response	Eles são enviados para informar ao cliente que a primeira parte da solicitação foi aceita e que ele deve continuar enviando o restante da solicitação. Esses códigos não são muito comuns.
200 até 299 - Sucesso	Esse intervalo de códigos de status é usado para informar ao cliente que sua solicitação foi bem-sucedida
300 até 399 - Redirecionamento	Eles são usados para redirecionar a solicitação do cliente para outro recurso. Isso pode ser para uma página da web diferente ou para um site totalmente diferente.
400 até 499 - Erro do cliente	Usado para informar ao cliente que houve um erro em sua solicitação.
500 até 599 - Erro do servidor	Isso é reservado para erros que ocorrem no lado do servidor e geralmente indicam um problema grave com o servidor que trata a solicitação.

Existem muitos códigos de status HTTP diferentes e isso não inclui o fato de que os aplicativos podem definir os seus próprios:

Código	Descrição
200 - OK	A solicitação foi concluída com sucesso.
201 - Created	Um recurso foi criado; por exemplo, um novo usuário ou uma nova postagem no blog
301 - Moved permanently	Isso redireciona o navegador do cliente para uma nova página da web ou informa aos mecanismos de pesquisa que a página foi movida para outro lugar e deve procurar lá.
302 - Found	Semelhante ao redirecionamento permanente acima, mas como o nome sugere, esta é apenas uma mudança temporária e pode mudar novamente em um futuro próximo.
400 - Bad request	Isso informa ao navegador que algo estava errado ou faltando em sua solicitação. Às vezes, isso pode ser usado se o recurso do servidor web solicitado espera um determinado parâmetro que o cliente não enviou.
401 - Not authorized	No momento, você não tem permissão para visualizar este recurso até que tenha autorizado o aplicativo da web, geralmente com um nome de usuário e uma senha.
403 - Forbidden	Você não tem permissão para visualizar este recurso, esteja conectado ou não.
405 - Method not allowed	O recurso não permite essa solicitação de método, por exemplo, você envia uma solicitação GET para o recurso /create-account quando ele esperava uma solicitação POST.
404 - Page not found	A página/recurso que você solicitou não existe.
500 - Internal Service error	O servidor encontrou algum tipo de erro em sua solicitação que não sabe como tratar adequadamente
503 - Service unavailable	Este servidor não pode atender sua solicitação porque está sobrecarregado ou fora do ar para manutenção

3.Cabeçalhos HTTP

Cabeçalhos são bits adicionais de dados que você pode enviar ao servidor web ao fazer solicitações. Embora nenhum cabeçalho seja estritamente necessário ao fazer uma solicitação HTTP, você achará difícil visualizar um site corretamente. Esses são os cabeçalhos enviados do cliente (geralmente seu navegador) para o servidor.

- **Host:** Alguns servidores da web hospedam vários sites, portanto, ao fornecer os cabeçalhos do host, você pode informar qual deles precisa, caso contrário, receberá apenas o site padrão do servidor.

- **User-agent:** Este é o software do seu navegador e o número da versão, informando ao servidor da web que o software do seu navegador ajuda a formatar o site corretamente para o seu navegador e também alguns elementos de HTML, JavaScript e CSS estão disponíveis apenas em determinados navegadores.
- **Content-length:** Ao enviar dados para um servidor web, como em um formulário, o comprimento do conteúdo informa ao servidor web quantos dados esperar na solicitação web. Dessa forma, o servidor pode garantir que nenhum dado esteja faltando.
- **Accept-encoding:** Informa ao servidor web quais tipos de métodos de compactação o navegador suporta para que os dados possam ser reduzidos para transmissão pela Internet.
- **Cookie:** Dados enviados ao servidor para ajudar a lembrar suas informações

Esses são os cabeçalhos enviados do servidor para o cliente após uma solicitação

- **Set-cookie:** Informações para armazenar que são enviadas de volta ao servidor web em cada solicitação
- **Cache-control:** Quanto tempo armazenar o conteúdo da resposta no cache do navegador antes de solicitá-la novamente
- **Content-type:** Isso informa ao cliente que tipo de dados está sendo retornado, ou seja, HTML, CSS, JavaScript, Imagens, PDF, Vídeo, etc. Usando o cabeçalho do tipo de conteúdo, o navegador sabe como processar os dados
- **Content-encoding:** Qual método foi usado para compactar os dados para torná-los menores ao enviá-los pela internet