****

**Visão**

Com a crescente demanda sobre Tecnologias, percebemos que muitas pessoas apesar de buscarem informações, não possuem fontes que queiram realmente passar o conhecimento da maneira como ela deve ser, livre e com embasamento técnico que permita ser aplicado e utilizado quando necessário, além de serem testados em sua criação, tornando esta informação útil e confiável.

**Missão**

O Laboratório foi criado com a intenção de buscar e disseminar o conhecimento de uma maneira clara e objetiva, de forma gratuita, auxiliando na evolução dos membros e da sociedade na qual estas informações são compartilhadas, buscando o crescimento de todos os envolvidos nesta criação de valores.



Caso você pense que com a leitura dos materiais da How2Security, você irá se tornar um Cracker capaz de invadir sistemas, se você espera encontrar aqui scripts infalíveis para invasão e, a partir deles, sair por aí invadindo computadores, essa não é a leitura indicada. Indicamos, sim a leitura do Código Penal (Lei 2.848/1940), principalmente a Lei Carolina Dickmann (Lei 12.737/2012), nos Artigos 154-A e 154-B.

*154-A Invadir dispositivo informático alheio, conectado ou não à rede de computadores, mediante violação indevida de mecanismo de segurança e com o fim de obter, adulterar ou destruir dados ou informações sem autorização expressa ou tácita do titular do dispositivo ou instalar vulnerabilidades para obter vantagem ilícita:*

*Pena – Detenção, de 3 meses a 1 ano, e multa*

Este material é um conjunto de informações compiladas de documentos e ferramentas do Mundo Underground testadas em ambiente de laboratório na nossa intranet. Desta forma, todo conhecimento aqui condensado é tangível, assim como as orientações das contramedidas.

Dessa forma, esperamos ter sido bem claros que, em momento algum, estamos com a pretensão de ensinar a você como se tornar um invasor. Estaremos sim, mostrando muitas das técnicas utilizadas pelos crackers e, em alguns casos, pelos scripts kiddies, para que você, como administrador de redes, seja capaz de identificá-las em tempo hábil para se defender, antes que alguém com desejos menos nobres ô faça por você.

Assim sendo, todo o conteúdo dessa literatura tem apenas o objetivo didático de informar e preparar os administradores de redes dos novos tempos. Em momento algum nos responsabilizamos pelo mau uso desse conhecimento ou por danos causados em seu equipamento ou de terceiros, assim como também não somos responsáveis pelos códigos e ferramentas aqui citados.

Sandro Melo

Adaptado por Wellington Silva aka Well

**0 – Protocolo HTTP (HyperText Transfer Protocol)**

O **Protocolo HTTP** é um protocolo de aplicação utilizado principalmente para acessar dados da Word Wide Web. HTTP funciona como uma combinação de FTP e SMTP. Ele é semelhante ao FTP porque transfere arquivos e usa o serviço do TCP. Entretanto, é muito mais simples do que o FTP. Não há nenhuma conexão de controle separada somente dados são transferidos entre o cliente e o servidor.

O HTTP é semelhante ao SMTP, porque os dados transferidos entre cliente e servidor parecem mensagens SMTP. Além disso, o formato das mensagens é controlado por cabeçalhos como o MIME. Ao contrário do SMTP, as mensagens HTTP não se destinam a serem lidas por seres humanos elas são lidas e interpretadas pelo servidor HTTP e pelo cliente HTTP (browser). As mensagens SMTP são armazenadas e encaminhadas, mas as mensagens HTTP são transmitidas imediatamente. Os comandos do cliente ao servidor são incorporados em uma mensagem de pedido. O conteúdo do arquivo solicitado ou de outras informações é incorporado em uma mensagem de resposta. O HTTP usa os serviços do TCP na porta conhecida 80.

**1 – Fundamentos do HTTP**

A documentação do protocolo HTTP pode ser encontrada nas RFCs 1945 (versão HTTP 1.0), 2616 (versão HTTP 1.1) e 7540 (versão HTTP 2.0). Porém os principais fundamentos do HTTP são:

* **HTTP é Ligtness (Texto Puro)** 🡪 Podemos ver todas as mensagens HTTP que atravessam a rede em texto puro. É muito fácil capturar o conteúdo do HTTP, e é fácil para os seres humanos interpretarem e entenderem. Por isso, o HTTP é muito fácil de simular as solicitações. Independente do aplicativo usual ser um browser da web, um flash player ou um leitor de PDF, ou qualquer outra coisa, podemos simular essas solicitações usando qualquer cliente que desejemos.
* **HTTP é Stateless (Sem Estado)** 🡪 O próprio protocolo HTTP não tem noção de estado, isto é, uma conexão não tem nenhuma relação com nenhuma outra conexão. Se clicarmos em um link agora e depois em outro link no mesmo host, o servidor não tem nenhum conceito de que a mesma pessoa tenha feito essas duas solicitações.
* **HTTP é Client/Server (Cliente/Servidor)** 🡪 No qual um cliente, usando um browser (navegador), pode acessar um serviço usando um servidor. O serviço fornecido e distribuído por muitos locais, chamados sites. Cada site contém um ou mais documentos, referidos como páginas web. Cada página web pode conter um link para outras páginas no mesmo site ou em outros. Toda a solicitação é feita pelo cliente para um servidor web e o servidor web responde para o cliente. Não é possível que um servidor decida que um cliente precisa de alguma informação e então ele se conecta no cliente para enviar esses dados.

A principal diferença entre a versão 1.0 e a versão 1.1 está na persistência na conexão da versão 1.1. Observe no exemplo abaixo:

root@kali-well:~scripts# nc www.securitytube.net 80

**GET / HTTP/1.0**

**Host: www.securitytube.net**

HTTP/1.0 200 OK

Content-Type: text/html; charset=utf-8

Cache-Control: no-cache

X-Cloud-Trace-Context: 8c43fddf01b345e4aadefa1af5ea20ab

Date: Fri, 24 Feb 2017 03:54:57 GMT

Server: Google Frontend

Content-Length: 31787

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

<head>

--==[ Resumido ]==--

</body>

</html>

root@kali-well:~scripts#

Na versão 1.0, assim que o documento foi mostrado a conexão TCP é desfeita. Agora vamos olhar a versão 1.1.

root@kali-well:~scripts# nc www.securitytube.net 80

**GET / HTTP/1.1**

**Host: www.securitytube.net**

HTTP/1.1 200 OK

Content-Type: text/html; charset=utf-8

Cache-Control: no-cache

X-Cloud-Trace-Context: 5191d6f4b56b1e282a2f87d106be6679

Date: Fri, 24 Feb 2017 03:55:15 GMT

Server: Google Frontend

Content-Length: 31787

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

<head>

--==[ Resumido ]==--

</body>

</html>

**GET / HTTP/1.1**

**Host: www.securitytube.net**

HTTP/1.1 200 OK

Content-Type: text/html; charset=utf-8

Cache-Control: no-cache

X-Cloud-Trace-Context: ed542d3cab9b45b035c817c22c9b05e5

Date: Fri, 24 Feb 2017 03:55:29 GMT

Server: Google Frontend

Content-Length: 31787

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

<head>

--==[ Resumido ]==--

</body>

</html>

**GET / HTTP/1.1**

**Host: www.securitytube.net**

**Connection: close**

HTTP/1.1 200 OK

Content-Type: text/html; charset=utf-8

Cache-Control: no-cache

X-Cloud-Trace-Context: e6d6431ee5bd9162706a10c2d7f77245

Date: Fri, 24 Feb 2017 03:55:48 GMT

Server: Google Frontend

Content-Length: 31787

Connection: close

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

<head>

--==[ Resumido ]==--

</body>

</html>

root@kali-well:~scripts#

Desta vez a conexão *TCP* só foi fechada quando passamos no cabeçalho a opção *Connection* como *Close*.

**1.1 – URL (Uniform Resource Locator)**

Um cliente que queira acessar uma página da web precisa do endereço. Para facilitar o acesso a documentos distribuídos por todo o mundo, o protocolo HTTP usa localizadores. O URL é um padrão para especificar qualquer tipo de informação na Internet. O URL define seis informações: protocolo, usuário, senha, computador host, porta e caminho.

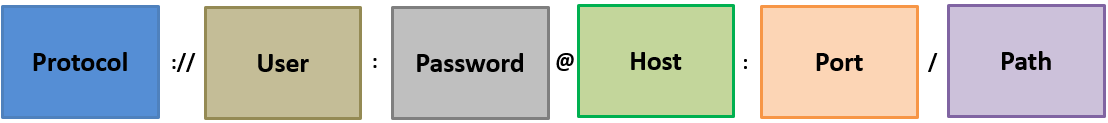


Figura 01 - URL

* **Protocol (Protocolo)** 🡪 É o programa cliente servidor usado para recuperar o documento. Muitos protocolos diferentes podem recuperar um documento, entre eles estão **Gopher**, **FTP**, **HTTP**, **News**, etc. O mais comum atualmente é o HTTP.
* **User (Usuário)** 🡪 Podemos passar um ID de usuário diretamente na URL. Este campo na URL é opcional e dependendo da aplicação, ele pode ser case sensitive (sensível à caixa das letras).
* **Password (Senha)** 🡪 Além do ID de usuário também podemos passar a senha deste usuário. Esse campo também é opcional e dependendo da aplicação pode ser case sensitive.
* **Host (Computador)** 🡪 É o computador no qual a informação está localizada, embora o nome do computador possa ser um apelido. Normalmente, as páginas web são armazenadas em computadores e estes recebem apelidos que começam com os caracteres **“www”**. Contudo, isso não é obrigatório, pois o host pode ter qualquer nome dado ao computador que contém a página da web.
* **Port (Porta)** 🡪 Facultativamente, o URL pode conter o número de porta do servidor. Se a porta for incluída, ela é inserida entre o host e o caminho, e é separada do host por dois-pontos (:).
* **Path (Caminho)** 🡪 É o nome de caminho do arquivo em que a informação está localizada.

**1.2 – Cookies**

Como já visto, o HTTP é uma entidade sem estado. Um cliente envia um pedido; um servidor responde. Sua relação está terminada. O projeto original **da www (Word Wide Web)**, de recuperar documentos publicamente disponíveis, se ajusta exatamente a esse propósito. Atualmente, a web tem outras funções: algumas estão listadas a seguir:

* Alguns sites precisam permitir o acesso apenas a clientes registrados;
* Sites sendo usados como lojas eletrônicas que permitem aos usuários navegar na loja, selecionar os itens desejados, coloca-los em um carrinho eletrônico e, no fim, pagar com um cartão de crédito;
* Alguns sites são usados como portais, onde os usuários utilizam serviços personalizados;
* Alguns sites são de bancos (Internet Bank), onde os usuários utilizam para fazer atividades bancárias, como pagar contas, transferir fundos, aplicar dinheiro, etc;
* Alguns sites são apenas anúncios.

Para esses propósitos, foi desenvolvido o mecanismo de cookie.

A criação e o armazenamento do cookie dependem da implementação, contudo, o princípio é o mesmo:

1. Quando um servidor recebe um pedido de um cliente, ele armazena informações sobre o cliente em um arquivo ou em uma string. As informações podem incluir o nome do domínio do cliente, o conteúdo do cookie (informações sobre o cliente reunidas pelo servidor, como nome, número de registro, etc.), uma marcação de horário e outras informações, dependendo da implementação;
2. O servidor inclui o cookie na resposta que envia ao cliente (Set-Cookie);
3. Quando o cliente receber a resposta, o browser armazena o cookie no diretório de cookie, que é classificado pelo nome do servidor do domínio.

Quando um cliente envia um pedido a um servidor, o browser examina o diretório de cookies para ver se consegue encontrar um cookie enviado por esse servidor. Se for encontrado, o cookie é incluído no pedido. Quando o servidor recebe o pedido, ele sabe que é de um cliente antigo, e não de um novo cliente. Note que o conteúdo do cookie nunca é lido pelo browser nem revelado ao usuário. Trata-se de um cookie feito pelo servidor e consumido pelo servidor. Vamos ver agora como um cookie é usado para os cincos propósitos mencionados anteriormente:

1. O site que restringe o acesso apenas a clientes registrados envia um cookie para o cliente quando ele se registra pela primeira vez. Para todo acesso repetido, somente os clientes que enviam o cookie apropriado são permitidos.
2. Uma loja eletrônica (e-commerce) pode usar um cookie para seus clientes. Quando um cliente seleciona um item e o insere em um caminho, um cookie com informações sobre o item, como seu número e preço unitário, é enviado ao browser. Se o cliente selecionar um segundo item, o cookie é atualizado com a nova informação de seleção e assim por diante, Quando um cliente termina as suas compras e quiser pagar e sair, o último cookie é recuperado e o preço total é calculado.
3. Uma Internet Bank pode usar um cookie para identificar seus clientes. Quando um cliente acessa o Internet Bank e insere seus dados no formulário de autenticação como agência, número da conta, nome do cliente, etc, um cookie com informações sobre o cliente, como os dados inserido pelo cliente, assim como, tempo de expiração do cookie, características de uso do cliente dentro do portal do banco para personalizar a página inicial com os serviços que esse cliente mais utiliza quando on-line etc.
4. Um portal da web usa o cookie de maneira semelhante. Quando um usuário seleciona suas páginas prediletas, um cookie é produzido e enviado. Se o site for acessado novamente, o cookie é enviado ao servidor para mostrar o que o cliente está procurando.
5. Um cookie também pode ser usado por agências de propaganda. Uma agência de propaganda pode coletar faixas de anúncios em algum site principais que é frequentemente visitado pelo usuário. A agência de propaganda fornece apenas um URL com endereço da faixa, em vez da própria faixa. Quando um usuário visita o site principal e clica no ícone de uma empresa anunciada, é enviado um pedido para a agência de propaganda. A agência de propaganda envia a faixa, mas também inclui um cookie com o ID do usuário. Todo uso futuro das faixas é adicionado no banco de dados, que traça o perfil do comportamento do usuário na web. A agência de propaganda compila os interesses do usuário e pode vender essas informações a outros.

**1.3 – Documentos da Web**

Os documentos na WWW podem ser agrupados em três amplas categorias: estáticos, dinâmicos e ativos. A categoria é baseada no momento em que o conteúdo do documento é determinado.

**1.3.1 – Documentos Estático**

Os documentos estáticos são documentos de conteúdo fixo, criados e armazenados em um servidor. O cliente só pode obter uma cópia do documento, em outras palavras, o conteúdo do arquivo é determinado quando o arquivo é criado, e não quando ele é usado. Evidentemente o conteúdo pode ser alterado no servidor, mas no cliente não pode ser alterado. Quando um cliente acessar o documento, uma cópia é enviada. O usuário pode então usar um programa de navegação para exibir o documento.

O HTML (HyperText Markup Language) é uma linguagem para a criação de páginas web. O termo linguagem de marcação é proveniente do setor editorial, onde eles marcavam os textos que eles queriam destacar. Do mesmo modo, os dados de uma página da web são formatados para interpretação feita pelo browser.

**1.3.2 – Documentos Dinâmicos**

Um documento dinâmico é criado pelo servidor web quando o navegador solicita o documento. Quando chega um pedido, o servidor da web executa um programa aplicativo ou um script que cria o documento dinâmico. O servidor retorna a saída (um pequeno texto) do programa ou do script como uma resposta ao browser que solicitou o documento. Como um documento novo é criado para cada pedido, o conteúdo de um documento dinâmico pode variar de um pedido para o outro.

A CGI (Common Gateway Interface) é uma tecnologia que cria e manipula documentos dinâmicos. A CGI é um conjunto de padrões que definem como um documento dinâmico é escrito, como os dados são introduzidos no programa e como o resultado de saída é usado.

A CGI não é uma linguagem nova, em vez disso, ela permite aos programadores usar uma de várias linguagens, como C, C++, Bourne Shell, Korn Shell, C Shell, Perl, etc. A única coisa que a CGI define é um conjunto de regras e termos que o programador deve seguir.

Outras tecnologias de documentos dinâmicos são: JSP, PHP, ASP, ASP.Net, Cold Fusion entre outras.

**1.3.3 – Documentos Ativos**

Tem algumas aplicações que precisam executar um programa ou script nas instalações do cliente. Eles são chamados de documentos ativos. Quando um navegador solicita um documento ativo, o servidor envia uma cópia do documento ou script. Então o documento é executado nas instalações do cliente (browser).

Uma maneira de criar documento ativo é usando Applets Java. Java é uma combinação de programação de alto nível, ambiente de tempo de execução e biblioteca de classes que permite a um programador escrever um documento ativo e um browser executá-lo. Outra forma de construir documentos ativos é utilizando JavaScript que é uma linguagem de script interpretada e executada no cliente.

**2 – Transação HTTP**

Embora o HTTP use os serviços do TCP, ele próprio é um protocolo state-less (sem estado). O cliente inicializa a transação enviando uma mensagem de pedido. O servidor responde enviando uma resposta.

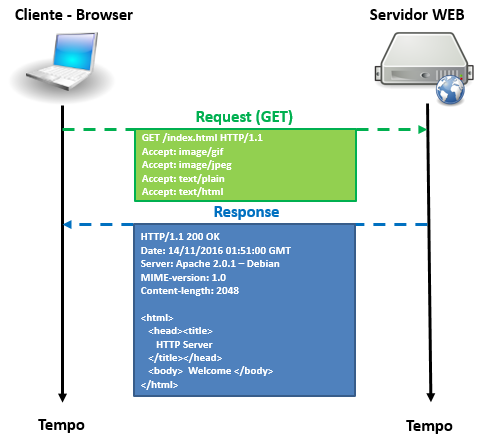


Figura 02 – Troca de Mensagens no HTTP

O formato da mensagem de pedido e de resposta são semelhantes; ambos são mostrados logo abaixo:

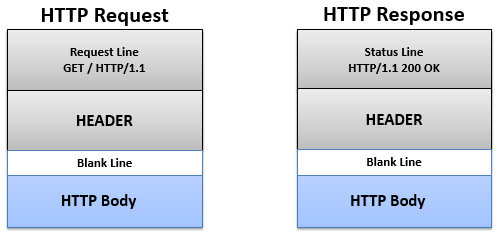


Figura 03 – Mensagens HTTP

Uma mensagem de pedido consiste em uma linha de pedido (request line) um cabeçalho (header) e as vezes um corpo (blank line e http body). Uma mensagem de resposta consiste em uma linha de estado (status line) um cabeçalho (header) e, às vezes, um corpo (blank line e http body).

O formato do protocolo HTTP Request:

* **Request Line (Linha de Pedido)** 🡪 A primeira linha em uma mensagem de pedido é chamada de Request Line e temos três campos nela:
  + **Request Type (Tipo de Pedido)** 🡪 Este campo é usado na mensagem de pedido. Na versão 1.1 do HTTP vários tipos são definidos. O tipo é classificado em métodos confirme pode ser visto na Tabela 01 adiante;
  + **URL (Uniform Resource Locator)** 🡪 É um tipo especial de URI (Uniform Resource Identifier). Ela indica a localização de algo que estamos tentando manipular através do HTTP. Por exemplo, ***http://www.how2security.com.br/dir/arquivo1.html***, onde, queremos o ***arquivo1.html*** que se encontra dentro do diretório ***dir*** no host ***www.how2security.com.br*** utilizando o protocolo http;
  + **Version (Versão)** 🡪 É a versão do protocolo HTTP temos as versões 1.0, 1.1 (a mais utilizada) e para ser lançada a 2.0 (até 2016 estava em teste).
* **Header (Cabeçalho)** 🡪 O header troca informações adicionais entre o cliente e o servidor. Por exemplo, o cliente pode solicitar que um arquivo seja enviado em um formato especial ou o servidor pode enviar informações extras sobre o arquivo. Cada linha do header tem um nome, dois-pontos, um espaço e um valor. Uma linha do header pertence a uma de quatro categorias: General Header, Request Header, Response Header ou Entity Header. Uma mensagem de pedido pode conter somente os headers General, Request ou Entity. Por outro lado, uma mensagem de resposta pode conter somente os headers General, Response e Entity.
  + **General Header (Cabeçalho Geral)** 🡪 O cabeçalho geral fornece informações gerais sobre a mensagem e pode estar presente em um pedido e em uma resposta. A Tabela 03, lista algumas linhas e suas descrições;

|  |  |
| --- | --- |
| Header | Description |
| Cache-Control | Especifica informações sobre o uso de cache. |
| Connection | Mostra se a conexão deve ser fechada ou não. |
| Date | Mostra a data atual. |
| MIME-Version | Mostra a versão de MIME usada. |
| Upgrade | Especifica o protocolo de comunicação preferencial. |

Tabela 01 – General Header

* + **Request Header (Cabeçalho de Requisição)** 🡪 O cabeçalho de pedido pode estar somente em uma mensagem de pedido. Ele especifica a configuração do cliente e o formato do documento preferencial do cliente.

|  |  |
| --- | --- |
| Header | Description |
| Accept | Mostra o formato de mídia que o cliente pode aceitar |
| Accept-Charset | Mostra o conjunto de caracteres que o cliente pode manipular. |
| Accept-Encoding | Mostra o esquema de codificação que o cliente pode manipular. |
| Accept-Language | Mostra a linguagem que o cliente pode aceitar. |
| Authorization | Mostra quais permissões o cliente tem. |
| From | Mostra o endereço de e-mail do usuário. |
| Host | Mostra o host e o número de porta do cliente. |
| If-Modified-Since | Envia o documento se for mais recente do que a data especificada. |
| If-Match | Envia o documento somente se ele corresponde à tag dada. |
| If-None-Match | Envia o documento somente se ele não corresponde à tag dada. |
| If-Range | Envia apenas a parte do documento que está faltando. |
| If-Unmodified-Since | Envia o documento se não tiver sido alterado desde a data especificada, |
| Referrer | Especifica o URL do documento vinculado. |
| User-Agent | Identifica o programa cliente (browser). |
| Cookie | Envia cookies para o servidor. |
| Origin | É usado cross-domain nas requisições AJAX, para indicar o domínio de quem originou a requisição. |

Tabela 02 – Request Header

* + **Entiry Header (Cabeçalho de Entidade)** 🡪 O cabeçalho de entidade fornece informações sobre o corpo do documento. Embora este cabeçalho esteja presente principalmente em mensagem de resposta, algumas mensagens de pedido, como POST ou PUT, que contêm corpo, também usam este tipo de cabeçalho.

|  |  |
| --- | --- |
| Header | Description |
| Allow | Lista os métodos válidos que podem ser usados com um URL. |
| Content-Encoding | Especifica o esquema de codificação. |
| Content-Language | Especifica a linguagem. |
| Content-Length | Especifica o comprimento do documento. |
| Content-Range | Especifica o intervalo do documento. |
| Content-Type | Especifica o tipo do documento. |
| Tansfer-Type | Especifica qualquer codificação executada no corpo da mensagem para facilitar a transferência sobre HTTP. |
| Etag | Fornece uma tag de entidade. |
| Expires | Fornece a data e a hora em que o conteúdo poderá mudar. |
| Last-Modified | Fornece a data e a hora da última alteração. |
| Location | Especifica a localização do documento criado ou movido. |

Tabela 03 – Entity Header

* **Blank Line (Linha em Branco)** 🡪 Este sempre é colocado quando a mensagem contém um corpo.
* **HTTP Body (Corpo HTTP) 🡪** O corpo pode estar presente em uma mensagem de pedido ou de resposta. Normalmente, ele contém o documento a ser enviado ou recebido.

Os principais métodos suportados pelo HTTP estão listados na Tabela 04, abaixo:

|  |  |
| --- | --- |
| Método | Ação |
| GET | Solicita um documento do servidor. Em caso de conteúdo dinâmico utiliza a URI da mensagem para enviar os valores de parâmetros. |
| HEAD | Solicita informações sobre um documento, mas não o documento em si. |
| POST | Envia um documento do cliente para o servidor. Utiliza o corpo da mensagem para enviar os valores de parâmetros. |
| PUT | Envia um documento do servidor para o cliente. |
| TRACE | Ecoa o pedido recebido para diagnostico. |
| CONNECT | Converte a requisição de conexão para um túnel TCP/IP transparente, usualmente para facilitar a comunicação criptografada SSL (HTTPS) através de um proxy HTTP não criptografado. |
| OPTIONS | Pergunta sobre as opções disponíveis. |
| DELETE | Apaga um recurso especificado. |

Tabela 04 – Method HTTP

O formato do protocolo HTTP Response:

* **Status Line (Linha de Estado)** 🡪 Este campo é usado na mensagem de resposta. Ele tem três subcampos: Version, Status Code e Description of Status.
  + **Version (Versão)** 🡪 É a versão do protocolo HTTP temos as versões 1.0, 1.1 (a mais utilizada) e para ser lançada a 2.0 (até 2016 estava em teste);
  + **Status Code (Código de Estado)** 🡪 O campo de código de status é semelhante ao dos protocolos FTP e SMTP. Ele consiste em três dígitos. Embora o código no intervalo de 100 sejam apenas informativos, os códigos no intervalo de 200 indicam um pedido bem-sucedido. Os códigos no intervalo 300 redirecionam o cliente a outra URL, e os códigos no intervalo 400 indicam um erro nas instalações do cliente. Por fim, os códigos no intervalo 500 indicam um erro nas instalações do servidor. Listamos os códigos mais comuns na Tabela 02 adiante;
  + **Description Status (Descrição do Estado)** 🡪 Este campo é usado na mensagem de resposta. Ele explica o código de estado no formulário de texto, a Tabela 02 também fornece a descrição dos códigos de estado;
* **Header (Cabeçalho)** 🡪 O header troca informações adicionais entre o cliente e o servidor. Por exemplo, o cliente pode solicitar que um arquivo seja enviado em um formato especial ou o servidor pode enviar informações extras sobre o arquivo. Cada linha do header tem um nome, dois-pontos, um espaço e um valor. Uma linha do header pertence a uma de quatro categorias: General Header, Request Header, Response Header ou Entity Header. Uma mensagem de pedido pode conter somente os headers General, Request ou Entity. Por outro lado, uma mensagem de resposta pode conter somente os headers General, Response e Entity.
  + **General Header (Cabeçalho Geral)** 🡪 O cabeçalho geral fornece informações gerais sobre a mensagem e pode estar presente em um pedido e em uma resposta. A Tabela 03, lista algumas linhas e suas descrições;

|  |  |
| --- | --- |
| Header | Description |
| Cache-Control | Especifica informações sobre o uso de cache. |
| Connection | Mostra se a conexão deve ser fechada ou não. |
| Date | Mostra a data atual. |
| MIME-Version | Mostra a versão de MIME usada. |
| Upgrade | Especifica o protocolo de comunicação preferencial. |

Tabela 05 – General Header

* + **Response Header (Cabeçalho de Resposta)** 🡪 O cabeçalho de resposta pode estar presente somente em uma mensagem de resposta. Ele especifica a configuração do servidor e informações especiais sobre o pedido.

|  |  |
| --- | --- |
| Header | Description |
| Accept-Range | Mostra se o servidor aceita o intervalo solicitado pelo cliente. |
| Age | Mostra a idade do documento. |
| Public | Mostra a lista de métodos suportados. |
| Retry-After | Especifica a data após a qual o servidor está disponível. |
| Server | Mostra o nome do servidor e o número da versão. |
| Set-Cookie | Envia os Cookies para os clientes (browser), para ser enviado de volta para o servidor quando fizer as próximas requisições. |
| Access-Control-Allow-Origin | Indica se o recurso pode recuperar via cross-domain requisições AJAX. |
| Pragma | Passa o cache diretamente para o browser (por exemplo, no-cache). |
| WWW-Authenticate | É enviado na resposta quando houver um status code 401 para informar detalhes dos tipos de autenticações o servidor suporta. |

Tabela 06 – Response Header

* + **Entiry Header (Cabeçalho de Entidade)** 🡪 O cabeçalho de entidade fornece informações sobre o corpo do documento. Embora este cabeçalho esteja presente principalmente em mensagem de resposta, algumas mensagens de pedido, como POST ou PUT, que contêm corpo, também usam este tipo de cabeçalho.

|  |  |
| --- | --- |
| Header | Description |
| Allow | Lista os métodos válidos que podem ser usados com um URL. |
| Content-Encoding | Especifica o esquema de codificação. |
| Content-Language | Especifica a linguagem. |
| Content-Length | Especifica o comprimento do documento. |
| Content-Range | Especifica o intervalo do documento. |
| Content-Type | Especifica o tipo do documento. |
| Tansfer-Type | Especifica qualquer codificação executada no corpo da mensagem para facilitar a transferência sobre HTTP. |
| ETag | Fornece uma tag de entidade. |
| Expires | Fornece a data e a hora em que o conteúdo poderá mudar. |
| Last-Modified | Fornece a data e a hora da última alteração. |
| Location | Especifica a localização do documento criado ou movido. |

Tabela 07 – Entity Header

* **Blank Line (Linha em Branco)** 🡪 Este sempre é colocado quando a mensagem contém um corpo.
* **HTTP Body (Corpo HTTP) 🡪** O corpo pode estar presente em uma mensagem de pedido ou de resposta. Normalmente, ele contém o documento a ser enviado ou recebido.

Os principais códigos suportados pelo HTTP estão listados na Tabela 04, abaixo:

|  |  |
| --- | --- |
| Código | Descrição |
| 100 | Continue (A parte inicial do pedido foi recebido e o cliente pode continuar com seu pedido). |
| 101 | Switching Protocols (O servidor está consentindo em um pedido do cliente para trocar os protocolos definidos no cabeçalho de atualização). |
| 200 | OK (O pedido foi bem-sucedido). |
| 201 | Created (Um novo URL é criado). |
| 202 | Accepted (O pedido é aceito, mas não é posto em prática imediatamente). |
| 203 | Non-authoritative information |
| 204 | No content (Não há conteúdo no corpo). |
| 205 | Reset contente. |
| 206 | Partial contente. |
| 300 | Multiple choices |
| 301 | Moved permanently (O URL solicitado foi movido permanentemente). |
| 302 | Found (O URL solicitado foi movido temporariamente). |
| 303 | See other. |
| 304 | Not modified (O documento não foi modificado). |
| 305 | Use proxy. |
| 307 | Temporary redirect (O documento foi redirecionado temporariamente). |
| 400 | Bad Request (Há um erro de sintaxe no pedido). |
| 401 | Unauthorized (O pedido não tem autorização). |
| 401.1 | Unauthorized: Access is denied due to invalid credentials (Não autorizado devido as credenciais inválida). |
| 401.2 | Unauthorized: Access is denied due to server configuration favoring an alternative authentication method. |
| 401.3 | Unauthorized: Access is denied due to an Access Control List (ACL) set on the requested resource. |
| 401.4 | Unauthorized: Authorization failed by a filter installed on the web server. |
| 401.5 | Unauthorized: Authorization failed by an Internet Server Application Programming Interface/Common Gateway Interface (ISAPI/CGI) application. |
| 401.7 | Unauthorized: Access denied by URL authorization policy on the web server. |
| 402 | Payment required. |
| 403 | Forbidden (O serviço foi negado). |
| 403.1 | Forbidden: Execute access is denied. |
| 403.2 | Forbidden: Read access is denied. |
| 403.3 | Forbidden: Write access is denied |
| 403.4 | Forbidden: Secure Sockets Layer (SSL) is required to view this resource. |
| 403.5 | Forbidden: SSL 128 is required to view this resource. |
| 403.6 | Forbidden: IP address of the client has been rejected. |
| 403.7 | Forbidden: SSL client certificate is required. |
| 403.8 | Forbidden: Domain Name System (DNS) name of the client is rejected. |
| 403.9 | Forbidden: Too many clients are trying to connect to the web server. |
| 403.10 | Forbidden: Web server is configured to deny Execute access. |
| 403.11 | Forbidden: Password has been changed |
| 403.12 | Forbidden: Client certificate is denied access by the server certificate mapper. |
| 403.13 | Forbidden: Client certificate has been revoked on the web server. |
| 403.14 | Forbidden: Directory listing is denied on the web server. |
| 403.15 | Forbidden: Client access licenses have exceded limits on the web server. |
| 403.16 | Forbidden: Client certificate is ill-formed or is not trusted by the web server. |
| 403.17 | Forbidden: Client certificate has expired or is not yet valid. |
| 403.18 | Forbidden: Cannot execute requested URL in the current application pool. |
| 403.19 | Forbidden: Cannot execute CGIs for the client in this application pool. |
| 403.20 | Forbidden: Passport logon failed. |
| 404 | Not Found |
| 404.1 | File or directory not found: Web site not accessible on the requested port. |
| 404.2 | File or directory not found: Lockdown policy prevents this requested. |
| 404.3 | File or directory not found: Multipurpose Internet Mail Extensions (MIME) map policy prevents this request. |
| 405 | Method not allow (O método não é suportado nesse URL). |
| 406 | Not acceptable (O formato solicitado não é aceitável). |
| 407 | Proxy authentication required (É requisito a autenticação no proxy). |
| 408 | Request timeout (Tempo limite atingido). |
| 409 | Conflict. |
| 410 | Gone. |
| 411 | Length required. |
| 412 | Precondition failed. |
| 413 | Request entity too large. |
| 414 | Request Uniform Resource Identifier (URI) too large. |
| 415 | Unsupported media type. |
| 416 | Requested range not satisfiable. |
| 417 | Expectation failed. |
| 500 | Internal server error (Existe um erro, como uma falha, nas instalações do servidor). |
| 500.11 | Server error: Application is shutting down on the web server. |
| 500.12 | Server error: Application is busy restating on the web server (Aplicação está ocupada, reiniciando o web server). |
| 500.13 | Server error: Web server is too busy |
| 500.14 | Server error: Invalid application configuration on the server (Erro de configuração no servidor). |
| 500.15 | Server error: Direct requests for GLOBAL.ASA are not allowed. |
| 500.16 | Server error: Universal Naming Convention (UNC) authorization credentials incorrect. |
| 500.17 | Server error: URL authorization store cannot be found. |
| 500.18 | Server error: URL authorization store cannot be opened. |
| 500.19 | Server error: Data for this file is configured improperly in the metabase. |
| 500.20 | Server error: URL authorization scope cannot be found. |
| 500.100 | Internal server error: ASP error. |
| 501 | Not implemented (A ação solicitada não pode ser executada). |
| 502 | Bad gateway (O servidor que está atuando como gateway/proxy recebeu uma resposta inválida do outro servidor). |
| 503 | Service unavailable (O serviço está temporariamente indisponível, mas poderá ser solicitado no futuro). |
| 504 | Gateway timeout (O servidor que está atuando como gateway/proxy atingiu o tempo máximo). |
| 505 | HTTP version not supported |

Tabela 08 – Status HTTP

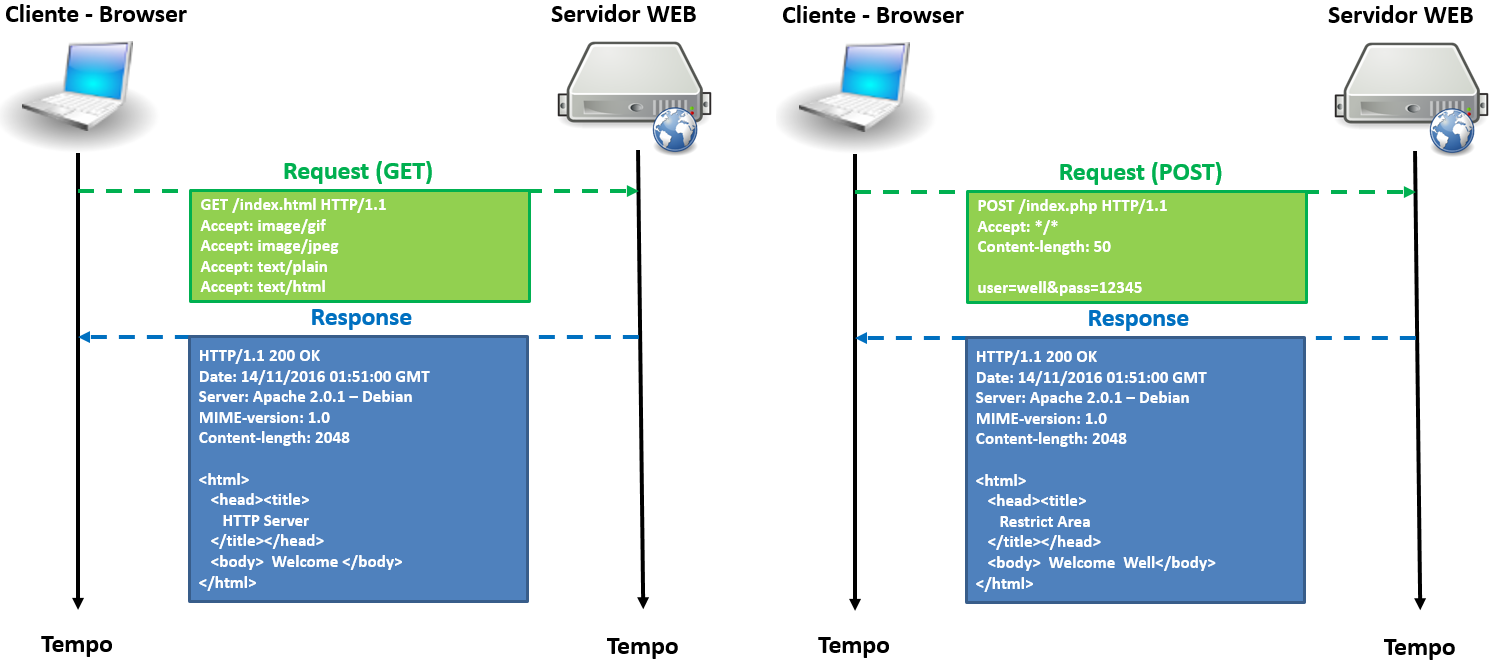


Figura 04 – Mensagens Trocas Entre Cliente/Servidor nos Métodos GET e POST

**3 – Servidor Proxy**

O **Protocolo HTTP** suporta servidores proxy. Um servidor proxy é um computador/aplicação que mantém cópias de respostas a pedidos recentes. O cliente HTTP envia um pedido ao servidor proxy. O servidor proxy verifica sua cache. Se a resposta não estiver armazenada na cache do servidor proxy, o servidor proxy envia o pedido ao servidor correspondente. As respostas recebidas são enviadas ao servidor proxy e armazenadas para pedidos futuros de outros clientes.

O servidor proxy reduz a carga no servidor original, diminui o tráfego e melhora a latência. Entretanto, para usar o servidor proxy, o cliente precisa ser configurado para acessar o proxy, em vez do servidor de destino.

**4 – REST-style URL**

O **REST (Representational State Transfer)** foi definido por Roy Thomas Fielding em sua dissertação de doutorado de 2000 “Archtectural Styles and the Design of Network-based Software Architectures” em UC Irvine. Fielding desenvolveu o estilo arquitetônico REST em paralelo com HTTP 1.1 de 1996-1999, com base no projeto existente do HTTP 1.0 de 1996.

O que pode nos interessar neste estilo de representação é a abstração nos nomes das variáveis que não são colocados na URL. Em vez disso, o REST-style URL é frequentemente usado para indicar seus parâmetros dentro de caminhos no URL, em vez da sequência utilizado no URL padrão do HTTP. Por exemplo, a URL padrão:

http://www.how2security.com.br/login?user=well&pass=123456

Corresponde a seguinte URL contendo parâmetros RESTfull:

http://www.how2security.com.br/login/well/123456

Desta forma temos os parâmetros passados como se fosse caminho na URL.

**5 – Referências**

**Referências Bibliográficas**

**[1]** Forouzan, Behrouz A. – Comunicação de dados e redes de computadores, 3º Ed, 2006, Porto Alegre, Bookman.

**[2]** Tanenbaum, Andrew S. – Redes de Computadores, 4º Ed, São Paulo, 2003, Editora Campus.

**[3]** Soares, Luiz Fernando G. – Redes de Computadores: das LANs, MANs, e WANs às redes ATM/Luiz Fernando Soares Gomes, Guido Lemos, Sergio Colcher, 1ª Ed, Rio de Janeiro, 1995, Editora Campus.

**[4]** Filippetti, Marco Aurélio – CCNA 4.0 – Guia Completo de Estudo, Florianópolis, 2006, Visual Books.

**[5]** RFC 1945, Disponivel em: <https://www.ietf.org/rfc/rfc1945.txt>. Acessado em: 14/11/2016.

**[6]** RFC 2616, Disponivel em: <https://www.ietf.org/rfc/rfc2616.txt>. Acessado em: 14/11/2016.

**[7]** RFC 7540, Disponivel em: <https://www.ietf.org/rfc/rfc7540.txt>. Acessado em: 14/11/2016.

**[8]** Forouzan, Behrouz A. – Protocolo TCP/IP, 3º Ed, 2008, São Paulo, McGraw-Hill.

**[9]** HOPE, Paco; WALTHER, Ben – Web Segura Guia de Testes e Soluções – Rio de Janeiro – 2009, Alta Books.

**[10]** STUTTARD, DAFYDD and PINTO, MARCUS – The Web Application Hacker’s Handbook, Indianapolis - Indiana, 2º Edition, Wiley Publishing Inc.

**[11]** HOPE, Paco; WALTHER, Ben – Web Segura Guia de Testes e Soluções – Rio de Janeiro – 2009, Alta Books.