Servidores HTTP, FTP, TELNET e SSH em Linux Debian-Based

Prof. Dr. Luiz Arthur Feitosa dos Santos
UTFPR – Universidade Tecnológica Federal do Paraná
DACOM – Departamento Acadêmico de Computação

Esse é um documento colaborativo da aula (notas de aula), há duas formas de editar:

I. **Envie um pedido de edição via Google e avise o professor (pode ser durante a aula mesmo);**
II. **Use o sistema de sugestão (já ativo), assim você pode editar/sugerir e depois o professor pode aceitar suas edições/sugestões.**

Ajudar na edição pode contar pontos na disciplina.

HTTP:

A seguir são apresentadas a instalação e algumas configurações para um servidor HTTP Apache em sistemas Linux Debian-based, tais como o Ubuntu.

Instalação:

```
#apt install apache2
```

Iniciando/parando Apache:

Iniciando:

```
# /etc/init.d/apache2 start
```

ou:

```
# apache2ctl start
```

ou no Debian/Ubuntu:

```
# systemctl start apache2
```

Parando:

```
# /etc/init.d/apache2 stop
```

ou:

```
# apache2ctl stop
```

ou no Debian/Ubuntu:

```
# systemctl stop apache2
```

Também é possível usar os mesmos comandos com a opção `restart` ao invés de `stop/start`.

Arquivos/diretórios de configuração:

No Debian o Apache possui uma configuração totalmente modular, de forma a trabalhar com módulos, configuração de servidores virtuais e outras diretivas da forma mais flexível possível, bem como permitir a automação de tarefas administrativas e torná-las o mais fácil possível. Isso divide a configuração do apache em vários arquivos/diretórios, sendo que os principais são (esses estão normalmente em `/etc/apache2`):

- **`apache2.conf`** – é o principal arquivo de configuração do Apache. Esse arquivo reúne as demais configurações do apache (que estão espalhadas nos outros arquivos/diretórios – ver a seguir);
- **`ports.conf`** – arquivo que determina as portas de rede e/ou hosts que responderão pelo servidor Apache;
- **`mods-available/`** - os módulos disponíveis para serem executados no Apache e assim fornecerem alguma funcionalidade extra, tal como o HTTPS. Tais módulos são estão necessariamente em execução no Apache, só disponíveis para serem executados;
- **`mods-enabled/`** - os módulos que realmente estão ativos (em execução) no Apache. Basicamente esse diretório tem um link para algum módulo do diretório `mods-available`;
- **`conf-available/`** - configurações disponíveis para serem executadas no Apache. Funciona tal como o mods-available.
- **`conf-enabled/`** - configurações que estão ativas no Apache, tal como o `mods-enabled`.
- **`sites-available/`** - configurações de sites ou hosts virtuais, disponíveis no Apache. Esse diretório é utilizado quando se deseja adicionar algum site novo, para isso basta criar um arquivo com o nome deste site e inserir as configurações pertinentes.
- **`sites-enabled/`** - são os sites que realmente estão ativos no Apache, tal como o `mods-enabled`.

Algumas configurações importantes do arquivo `apache2.conf`:

- **`ServerRoot "/etc/apache2"`** – diretório sob o qual as configurações, erros e arquivos de *logs* são mantidos;
- **`Timeout 300`** – Tempo que o servidor irá esperar por um evento antes de enviar uma resposta de falha, caso o evento não ocorra. São exemplos de espera: leitura e escrita em HD, transmissão TCP, etc.
- **`KeepAlive On`** – Se permite ou não conexões persistentes, nais quais várias operações podem ser feitas utilizando apenas uma conexão TCP. Em alguns casos isso dá um aumento de velocidade de aproximadamente 50% na conexão HTTP.
- **`User/Group`** – Nome do usuário e grupo de usuário que serão dadas permissões para a execução do Apache. Ou seja, se o usuário for o administrador o Apache terá permissão de root, o que é muito perigoso. Então, é recomendável que o Apache seja executado com um usuário comum e específico para sua execução.
- **`ErrorLog`** – Onde serão armazenados os arquivos de log de erros.
- **`LogLevel`** – Nível de log que será gravado, mais detalhado ou mais resumido. É possível configurar ou personalizar os logs do Apache.

Configurando as portas de rede do Apache no Debian-based:

Em distribuições Debian-based as portas são configuradas em um arquivo separado (`/etc/apache2/ports.conf`), em outras distros tais configurações estarão dentro do arquivo principal de configuração (`apache2.conf`). Por padrão o conteúdo do arquivo é algo como:

```
# cat /etc/apache2/ports.conf
Listen 80
<IfModule ssl_module>
Listen 443
</IfModule>
```

Na configuração padrão, anterior, o apache estará em execução na porta TCP/80 e caso o módulo SSL estiver ativo, o Apache também atenderá na porta TCP/443. Na configuração

de portas podemos alterar a porta padrão do Apache (trocando o número 80) ou adicionar outras portas inclusive relacionando-as com determinados IPs, exemplos:

- Executando o Apache nas portas TCP 80 e 8080:

```
Listen 80
Listen 8080
```

- Executando o Apache nas portas TCP 80 e 8080, mas especificando IPs que atenderão tais pedidos. Nesse exemplo, só será possível acessar a porta 80 a partir do *localhost* – não é possível acessar a porta TCP/80 via rede local ou Internet.

```
Listen 127.0.0.1:80
Listen 192.168.56.101:8080
```

Configuração de servidores virtuais:

No Apache há dois tipos de servidores virtuais, também chamados de hosts virtuais:

1. Servidores/hosts virtuais que se baseiam no endereço IP ou na porta de rede, sendo que nesse tipo deve ser alocado um endereço IP ou porta diferente para cada site.
2. Servidores/hosts virtuais que se baseiam no nome do domínio do servidor Web, já neste tipo, os sites são diferenciados pelo nome do host enviado pelo cliente HTTP (só funciona na versão 1.1 do HTTP, mas que é bem antiga). Este tipo de servidor/host virtual é mais indicada atualmente, devido a falta de endereços IPv4 disponíveis na Internet, contudo é mais complexo, principalmente se os hosts virtuais precisarem ser acessados via HTTPS (que será visto mais a frente).

Ainda quanto ao método (2) de hosts virtuais identificados por nome, o Apache possui um host virtual padrão, que é definido no arquivo `/etc/apache2/sites-enabled/000-default.conf`, tal host virtual é usado se não for encontrado nenhum outro host virtual que corresponda a requisição HTTP. A configuração de tal host virtual padrão é apresentada a seguir:

`# cat /etc/apache2/sites-enabled/000-default.conf`

```
<VirtualHost *:80>
 ServerName redes2
 ServerAdmin webmaster@localhost
 DocumentRoot /var/www/html
 ErrorLog ${APACHE_LOG_DIR}/error.log
 CustomLog ${APACHE_LOG_DIR}/access.log combined
 #Include conf-available/serve-cgi-bin.conf
```

```
</VirtualHost>
```

Na configuração anterior: o nome do servidor é `redes2`, o e-mail do administrador é `webmaster@localhost`, os arquivos disponibilizados pelo servidor virtual ficam armazenados no diretório `/var/www/html`, arquivos de logs ficam em `error.log` e `access.log`. Linhas que começam com `#` são comentários.

Criando dois hosts locais virtuais através de nomes:

Para criar um host virtual, basta criar um arquivo, normalmente com um nome que remete ao nome do host virtual no diretório `/etc/apache2/sites-available/`.
Por exemplo, vamos criar o host virtual que responde ao domínio www.tabajara.info:

```
# vi /etc/apache2/sites-available/www.tabajara.info.conf

<VirtualHost *:80>
      ServerName www.tabajara.info
      DocumentRoot /var/www/html/www.tabajara.info
</VirtualHost>
```

Depois dos passos anteriores, haverá um servidor virtual para www.tabajara.com disponível, contudo esse não estará habilitado, então é necessário criar um link simbólico para esse do diretório sites-available para sites-enabled. Isso é feito da seguinte forma:

```
# ln -l /etc/apache2/sites-available/www.tabajara.info.conf \
/etc/apache2/sites-enabled/
```

Por fim, reinicie o servidor Apache:

```
# apache2ctl restart
```

Atenção, para testar o host virtual é preciso utilizar nomes, para isso ou tal nome deve estar disponível no servidor DNS do cliente, ou (para testes) deve estar no arquivo `/etc/hosts` do cliente.

HTTPS:

Para instalar o HTTPS no Debian:

```
# a2enmod ssl
```

O comando a2enmod é um *script* que habilita módulos específicos do Apache. Na prática ele cria links simbólicos do diretório `mods-avaliable` para o `mods-enabled`. O Contrário do comando a2enmod é o a2dismod. Também, é possível utilizar as opções `-l` para listar módulos habilitados ou desabilitados e `-q` para questionar se um módulo está habilitado ou não. Tanto a2enmod quanto a2dismod estão presentes, a princípio, apenas em distros Debian-based. Ou seja, essa configuração do HTTPS só irá funcionar nestas distros.

```
# systemctl restart apache2
# make-ssl-cert generate-default-snakeoil --force-overwrite
# a2ensite default-ssl
# systemctl reload apache2
```

```
# vi /etc/apache2/sites-available/www.tabajara.info.conf
<VirtualHost *:80>
      ServerName www.tabajara.info
      DocumentRoot /var/www/html/www.tabajara.info
      Redirect / https://www.tabajara.info/
</VirtualHost>
```

```
<VirtualHost *:443>
      ServerName www.tabajara.info
      DocumentRoot /var/www/html/www.tabajara.info
</VirtualHost>
```

FTP

A seguir serão apresentados os comandos necessários para a instalação e configuração de um servidor FTP Proftpd em sistemas Debian-based. Também será apresentado o conceito do servidor de servidores xinetd, bem como configurar o Proftpd via xinetd.

Instalação:

Talvez seja necessário executar um update para atualizar a lista de pacotes do sistema operacional:

```
# apt update
```

para a instalação, propriamente dita execute:

```
# apt install proftpd
```

Iniciando/parando o Proftpd:

Inciando:

```
# /etc/init.d/proftpd start
ou

# systemctl start proftpd
```

Parando:

```
# /etc/init.d/proftpd stop

ou

# systemctl stop proftpd
```

É possível substituir stop/start nos comandos anteriores pelo parâmetro `restart`.

Arquivo de configuração:

O arquivo de configuração do Proftpd é normalmente o `/etc/proftpd/proftpd.conf`, as principais opções deste arquivo de configuração é:

- **`UseIPv6`** – habilita ou desabilita o suporte a IPv6;
- **`ServerName`** – nome do servidor;
- **`ServerType`** – configura se o servidor é standalone ou inetd/xinetd.
- **`DisplayLogin`** – indica um nome de arquivo que conterá uma mensagem que será apresentada para o usuário que logar no servidor FTP;

- **`DefaultRoot`** – não deixa o usuário sair de seu diretório home, isso pode ser muito importante por questões de segurança;
- **`Port`** - número da porta do FTP;
- **`MaxInstances`** – indica a quantidade máxima de conexões FTP. Isso pode evitar ataques DdoS.
- **`User/Group`** – usuário e grupo que irão executar o processo do Proftpd, não é recomendável utilizar o administrador, mas sim um usuário comum, por motivos de segurança.

Configurando acesso anônimo no Proftpd:

Para permitir acesso anônimo (sem login/senha) no Proftpd, é necessário editar o arquivo de configuração e adicionar/descomentar as seguintes linhas:

```
# vi /etc/proftpd/proftpd.conf
...
<Anonymous ~ftp>
 User ftp
 Group nogroup
 UserAlias anonymous ftp
 DirFakeUser on ftp
 DirFakeGroup on ftp
 RequireValidShell off
 MaxClients  10
 DisplayLogin  welcome.msg
 DisplayChdir .message
  <Directory *>
    <Limit WRITE>
      DenyAll
    </Limit>
  </Directory>
</Anonymous>
...
```

Então reinicie o proftpd:

```
# /etc/init.d/proftpd restart
```

Agora é possível acessar o servidor usando usuário/senha: `ftp/ftp` ou `anonymous/anonymous`. A maioria dos navegadores acessará sem pedir usuário e senha.

TELNET

Aqui serão apresentados os comandos para instalar um servidor TELNET.

Instalação:

```
#apt install telnetd
```

Não há muito o que configurar no telnetd, inclusive a sua inicialização normalmente está atrelada ao inetd ou xinetd, que é apresentado a seguir.

INETD/XINETD

O inetd ou xinetd são servidores de servidores. O xinet é um xinetd melhorado, mas basicamente eles fazem a mesma coisa. Inetd e xinetd são processos que ficam escutando determinadas portas de rede (configuradas em seus respectivos arquivos de configuração) e somente quando chega alguma requisição (pacotes) nestas portas é que eles executam os servidores que realmente respondem por aquelas portas. Assim, o inetd ou xinetd conseguem economizar recursos da máquina, já que os processos só são executados quando são realmente necessários, contudo isso pode gerar algum tipo de atraso na resposta do serviço solicitado, todavia esse atraso pode ser desprezível, além do que tanto o inetd quanto o xinetd podem prover mais segurança aos serviços que eles gerenciam.

Inetd

Servidor de servidores, mais antigo se comparado com o xinetd. No inetd todos os serviços são configurados em um único arquivo: `/etc/inetd.conf`.

Instalação:

```
# apt install openbsd-inetd
```

Alguns serviços como o telnetd são iniciados/configurados com o inetd, assim caso o telnetd já tenha sido instalado é bem provável que o inetd também já esteja instalado e não se faz necessário sua instalação.

Configuração do inetd:

O inetd utiliza um único arquivo para configuração. A seguir mostra-se como configurar o proftpd para iniciar com o inetd e não de forma standalone (que é o padrão).

```
#vi /etc/inetd.conf
telnet  stream  tcp nowait  telnetd /usr/sbin/tcpd  /usr/sbin/in.telnetd
ftp   stream tcp nowait root /usr/sbin/tcpd /usr/sbin/proftpd
```

No exemplo anterior, já havia uma linha para o telnet e foi adicionada a linha do proftpd (em negrito), nesta basicamente é informada em ordem: nome da porta (`ftp`), que o serviço é TCP (`stream tcp no wait`), o usuário que vai executar o serviço (`root`), o comando que será executado primeiro (`tcpd`) e o parâmetro (`proftpd`). O tcpd pode ser executado para dar mais segurança a execução dos serviços, no qual, por exemplo é possível configurar que hosts podem ou não usar o serviço (isso é feito com `/etc/hosts.allow` e `/etc/hosts.deny`).

Iniciando:

```
# /etc/init.d/openbsd-inetd start
```

ou

```
# systemctl start openbsd-inetd
```

Parando:

```
# /etc/init.d/openbsd-inetd start
```

ou

```
# systemctl restart openbsd-inetd
```

Se preferir também é possível utilizar a opção `restart`.

Restringindo serviços com `hosts.deny`

Usando o inetd com o tcpd é possível limitar/bloquear alguns serviços, tal como: para restringir que a rede 192.168.56.0/24 acesse o serviço do Proftpd edite o arquivo `/etc/hosts.deny` e inclua o seguinte conteúdo:

```
#vi /etc/hosts.deny
proftpd:192.168.56.0/24
```

Da mesma forma é possível liberar serviços através do arquivo `/etc/hosts.allow`.

Observação: não é necessário reiniciar o inetd para que as configurações de host.deny/allow passem a valer.

XINETD:

Servidor de servidores, considerado mais moderno, se comparado com o inetd. Utiliza um arquivo de configuração para cada serviço que será configurado via xinetd.

Atenção, não é recomendável executar o xinetd junto com inetd, então escolha apenas um entre xinetd e inetd.

Instalação:

```
# apt install xinetd
```

Configuração do xinetd:

A configuração geral do xinetd é feita pelo arquivo, mas basicamente não há muita configuração a se fazer-lá, a não ser que este arquivo inclui o diretório `/etc/xinetd.d`, que é o diretório no qual se faz a configuração dos serviços que serão mantidos pelo xinetd. Por exemplo, para configurar o proftdp via xinetd, é necessário criar o arquivo `/etc/xinetd.d/ftp`, que pode ser o seguinte conteúdo:

```
# vi /etc/xinetd.d/ftp
service ftp
{
 disable         = no
 flags           = REUSE
 socket_type     = stream
 wait            = no
 user            = root
 server          = /usr/sbin/proftpd
 server_args     = -c /etc/proftpd/proftpd.conf
 log_on_failure  += USERID
 no_access       = 192.168.57.0/24
 log_on_success  += PID HOST EXIT
 access_times    = 08:00-23:00
}
```

No exemplo anterior, o xinetd configura o proftpd para funcionar da seguinte forma: habilitar o servidor ftp (`disable = no`), utilizando o protocolo TCP (`socket_type = stream wait = no`), o comando a ser executado é o proftpd (`server = /usr/sbin/proftpd`), também são passados alguns argumentos ao proftpd (`server_args = -c /etc/proftpd/proftpd.conf`), tal servidor não pode ser acesso pela rede

192.168.57.0/24 (no_access = 192.168.57.0/24) e só pode ser acessado das 8:00hs às 23:00hs (`access_times = 08:00-23:00`).

Observação: o xinetd não faz usos arquivos hosts.allow/deny.

Iniciando:

```
# /etc/init.d/xinetd start
```

ou

```
# systemctl start xinetd
```

Parando:

```
# /etc/init.d/xinetd start
```

ou

```
# systemctl restart xinetd
```

Se preferir também é possível utilizar a opção `restart`.

SSH

O SSH foi desenvolvido para substituir o TELNET e o FTP, sanando os problemas de segurança (confidencialidade) desses. Assim, for necessário configurar ambientes Linux, o recomendável é utilizar o SSH, que irá trafegar tudo na rede através de mensagens criptografadas.

Instalação:

A maioria das distribuições Linux, principalmente as dedicadas a servidores, já terão por padrão o servidor SSH instalado e ativo. Contudo, se for realmente necessário instalar, execute:

#apt install openssh-server

Configuração:

O arquivo de configuração do servidor SSH é o `/etc/ssh/sshd_config`, as opções de configurações mais comuns são:

- `Port 22` - porta que será executada o SSH. Em um servidor real, principalmente exposto na Internet, é altamente recomendável mudar essa porta.
- `ListenAddress 0.0.0.0` - IPs que irão atender o serviços de SSH.
- `PermitRootLogin no` - permitir a login do root. Recomenda-se que o root ou qualquer usuário padrão não possa fazer login via SSH, por motivos de segurança.
- `PermitEmptyPasswords no` - Habilita ou desabilita o login de usuários que possuem senha em branco. Não é aconselhável, permitir que usuários que possuam senha em branco (sem senha), utilizem o servidor SSH. Se for necessário acessar sem senha, recomenda-se utilizar autenticação por chave criptográfica (isso não será abordado aqui).

Atenção, além do arquivo de servidor há o arquivo de configuração do cliente SSH (`/etc/ssh/ssh_config`) e é muito fácil confundi-los, pois só muda um “d” no arquivo.

Iniciando:

```
# /etc/init.d/ssh start
```
ou
```
# systemctl start ssh
```

Parando:

```
# /etc/init.d/ssh stop
```
ou
```
# systemctl stop ssh
```

Também é possível utilizar a opção `restart`.

Utilizando o SSH

Para acessar o servidor SSH, basta no cliente executar (é possível utilizar outros clientes, tal como o Putty):

```
$ ssh aluno@192.168.56.101
The authenticity of host '192.168.56.101 (192.168.56.101)' can't
be established.
ECDSA key fingerprint is
SHA256:mN4ViPROQ8QzTxUWoWoD8rsRDIhtCwA9oBe5slWTAMY.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.56.101' (ECDSA) to the list of
known hosts.
```

```
aluno@192.168.56.101's password:
Welcome to Ubuntu 18.04 LTS (GNU/Linux 4.15.0-47-generic x86_64)
...
aluno@redes2:~$
aluno@redes2:~$ ps
  PID TTY          TIME CMD
 5301 pts/1    00:00:00 bash
 5311 pts/1    00:00:00 ps

aluno@redes2:~$ exit
logout
Connection to 192.168.56.101 closed.
$
```

No exemplo anterior, acessamos o servidor 192.168.56.101, com usuário aluno. Como é o primeiro acesso, somos questionados se aceitamos a chave criptográfica do servidor (chave pública), isso é feito somente no primeiro acesso, depois não mais. Depois, caso acertamos a senha, estaremos dentro servidor e tudo que fizermos terá efeito no servidor (será executado pelo servidor), no caso foi executado o comando `ps` (para mostrar processos do servidor). Para sair do servidor e voltar a máquina do cliente, basta executar o comando `exit`.

Utilizando SFTP - FTP Seguro

Na maioria dos casos o servidor SSH dá acesso, por padrão, a um FTP seguro, chamado de SFTP. Esse pode ser acessado via interface gráfica (Konqueror, Filezilla e muitos gerenciadores de arquivo). Por exemplo, via linha de comando é possível utilizar o cliente sftp, tal como:

```
$ sftp aluno@192.168.56.101
aluno@192.168.56.101's password:
Connected to 192.168.56.101.
sftp> ls /etc/apache2
/etc/apache2/apache2.conf     /etc/apache2/conf-available
/etc/apache2/conf-enabled
/etc/apache2/envvars          /etc/apache2/magic
/etc/apache2/mods-available
/etc/apache2/mods-enabled     /etc/apache2/ports.conf
/etc/apache2/sites-available
/etc/apache2/sites-enabled
sftp> get /etc/apache2/apache2.conf
Fetching /etc/apache2/apache2.conf to apache2.conf
/etc/apache2/apache2.conf                                   100%
7224     7.1KB/s   00:00
sftp> exit
```

No exemplo, acessamos o servidor 192.168.56.101 com o usuário `aluno`, verificamos a existência do arquivo `apache2.conf`, então baixamos tal arquivo (`get /etc/apache2/apache2.conf`) e saímos do servidor. Após isso o arquivo em questão estará na máquina cliente.

Utilizando SCP - cópia segura

Também é possível fazer cópias entre hosts utilizando SSH, para isso é necessário que pelo menos uma máquina tenha um servidor SSH, mas há casos onde é necessário que duas tenham o servidor. Exemplos:

1. Copiar algo do servidor para o cliente:

   ```
   $ scp aluno@192.168.56.101:/etc/hosts  ~
   ```

   Neste caso o servidor é o `192.168.56.101`, o usuário é denominado `aluno`, e o arquivo a ser copiado é o `/etc/hosts`, esse será copiado para o diretório home do usuário do cliente (`~`).

2. Copiar algo do cliente para o servidor:

   ```
   $ scp /etc/apache2/apache2.conf aluno@192.168.56.101:/tmp
   ```

   Neste exemplo será copiado o arquivo `apache2.conf`, para o servidor `192.168.56.101`, utilizando a conta do usuário `aluno`, tal arquivo será copiado no diretório `/tmp`.

3. Copiando de um servidor SSH para outro:

   ```
   $ scp prof@192.168.56.102:/etc/resolv.conf aluno@192.168.56.101:~
   ```

   Aqui, o cliente copia o arquivo `resolv.conf` utilizando o usuário `prof` do servidor `192.168.56.102` para o diretório home do usuário `aluno` no servidor `192.168.56.101`. Ou seja, é uma operação envolvendo três máquinas, sendo um cliente (que inicia o comando) e dois servidores (um que tem o arquivo que será copiado e o outro que receberá tal cópia).

Criando túneis criptográficos com SSH:

O SSH permite criar túneis criptográficos do servidor SSH com o próprio servidor SSH ou com outros hosts. Isso é muito útil, por exemplo, para acessar redes locais que estão atrás de NATs. Para usar esse serviço só é necessário ter um usuário/senha válido no servidor SSH.

Exemplos de uso dos túneis SSH.

Atenção: Tais comandos devem ser digitados no cliente - não no servidor SSH:

1. Criando um túnel entre cliente e servidor SSH para acessar um site HTTP no próprio servidor:

   ```
   $ ssh -X -L 8080:127.0.0.1:80 aluno@192.168.56.101 -p 22
   ```

   No exemplo anterior é criado um túnel SSH do cliente com a máquina 192.168.56.101, utilizando o usuário aluno na porta TCP/80, tal porta estará disponível na máquina do cliente na porta TCP/8080. Atenção: A tela do login ficará ocupada com essa conexão, então é necessário abrir outro terminal, na máquina cliente. Assim, no cliente é possível acessar o túnel com o comando: `lynx 127.0.0.1:8080`.

2. Criando um túnel entre cliente e servidor SSH para que o servidor acesse uma máquina dentro de uma LAN conectada ao servidor:

   $ ssh -X -L 2223:172.16.1.194:23 aluno@172.16.1.195 -p 22

   Aqui, é criada um túnel do servidor SSH (aluno@172.17.2.195) para a máquina 172.16.1.194 na porta TCP/23. Tal conexão ficará disponível no cliente na porta TCP/2223 no 127.0.0.1. Então, para acessar o túnel basta o cliente, depois de ter conectado no SSH, digitar (em outro terminal) o comando `telnet 127.0.0.1 2223`.

Referências:

- https://archive.apache.org/dist/httpd/docs/httpd-docs-2.4.16.en.pdf
- https://debian-handbook.info/browse/pt-BR/stable/sect.http-web-server.html
- http://lrodrigo.sgs.lncc.br/wp/dicas/apache-ativando-o-protocolo-https-http-ssl/
- https://wiki.freephile.org/wiki/A2enmod