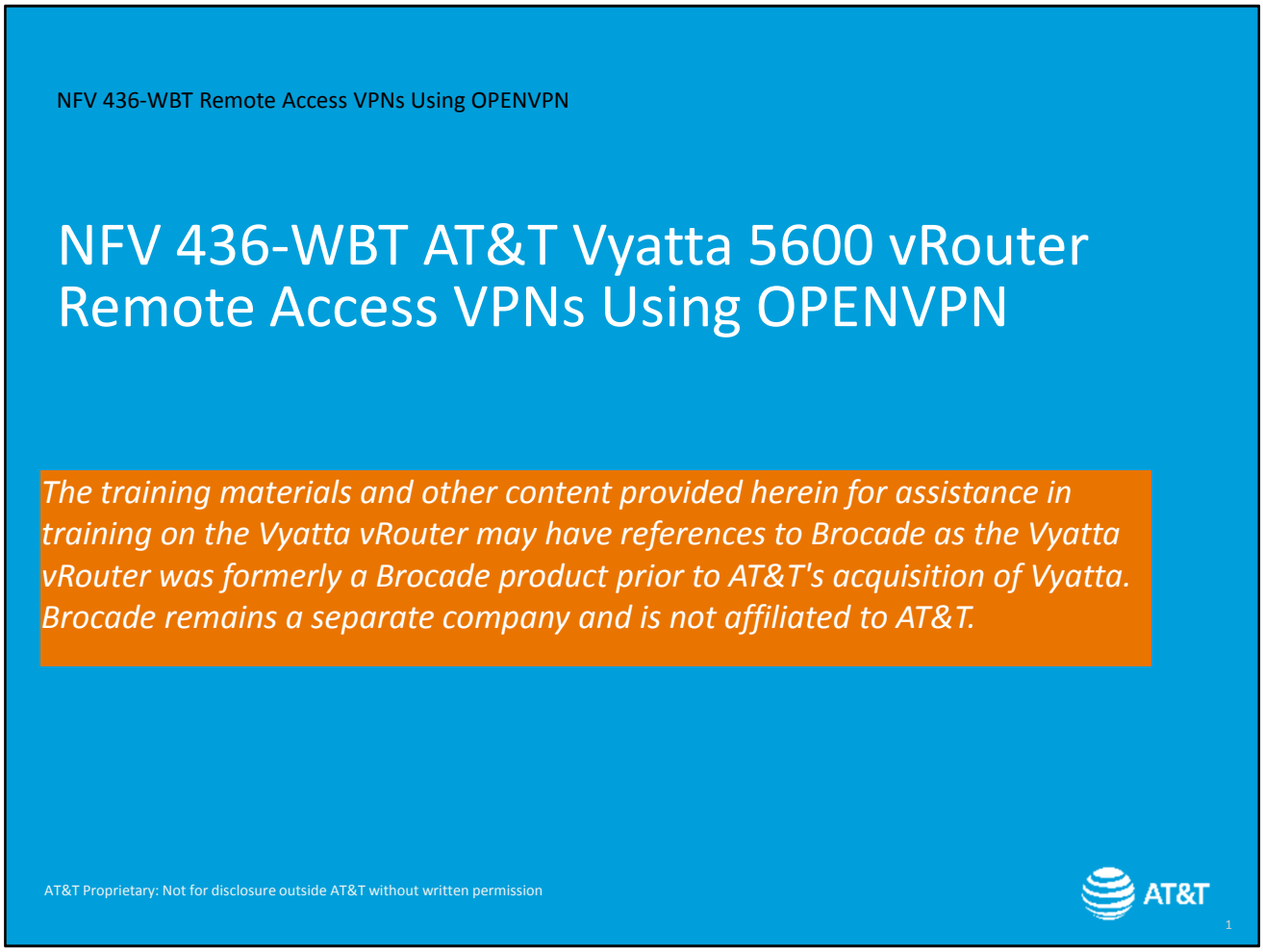


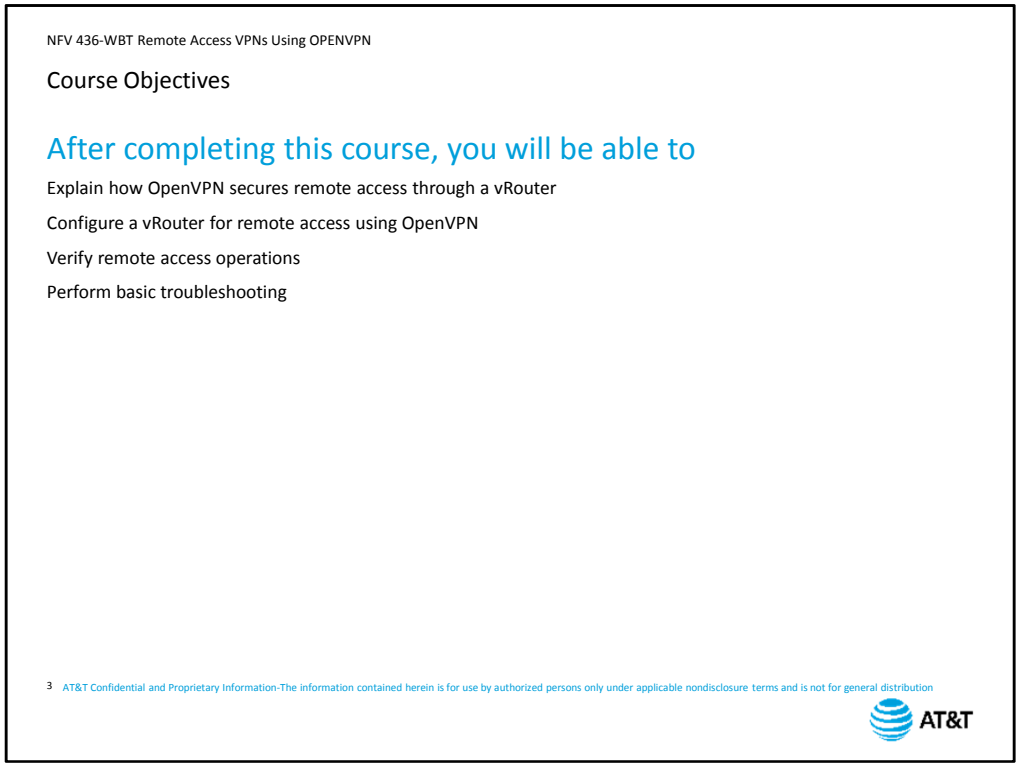
Welcome to the AT&T vRouter Dynamic Multipoint VPN course.

Legal Disclaimer

All or some of the products detailed in this presentation may still be under development and certain specifications may be subject to change. Nothing in this presentation shall be deemed to create a warranty of any kind.

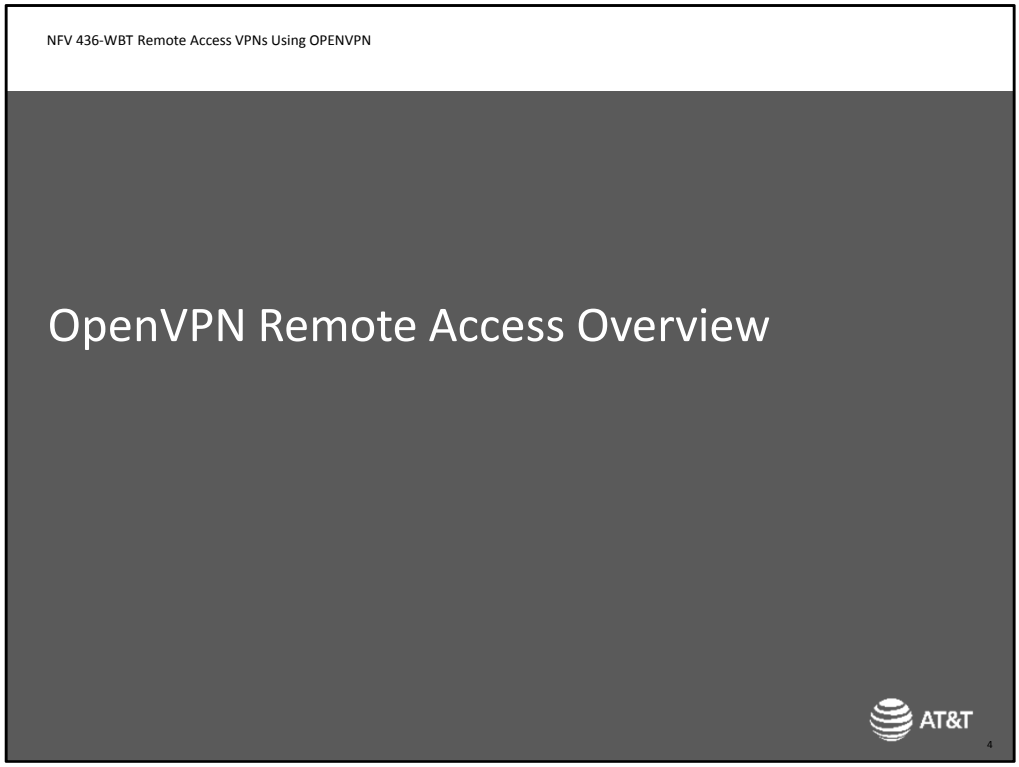
AT&T

Before we begin the course, please take a moment to read our legal disclaimer.

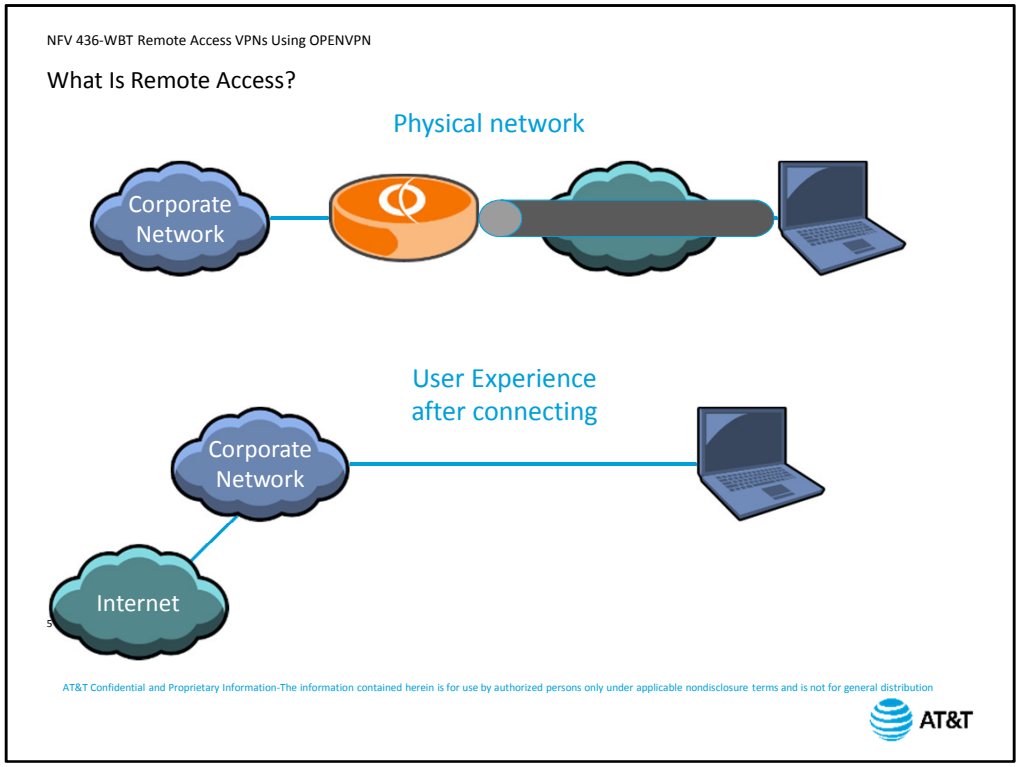


Welcome to the AT&T vRouter Remote Access Using OpenVPN course.
After completing this course, you will be able to:

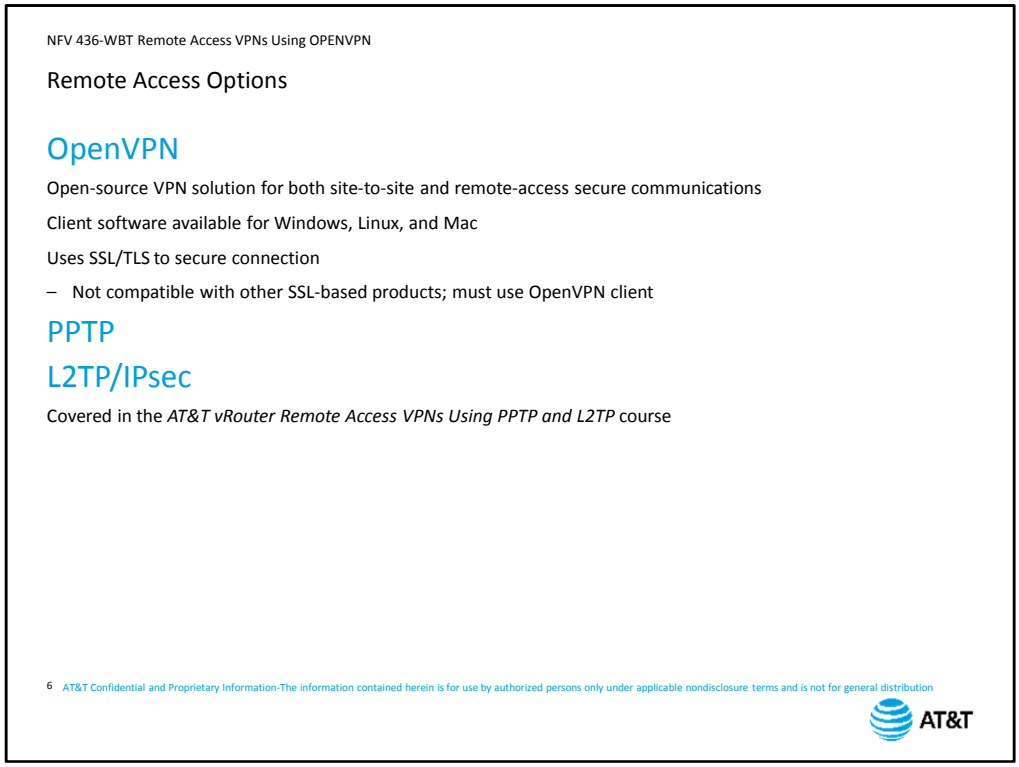
- Explain how OpenVPN secures remote access through a vRouter
- Configure the vRouter for remote access using OpenVP
- Verify remote access operations
- Perform basic troubleshooting



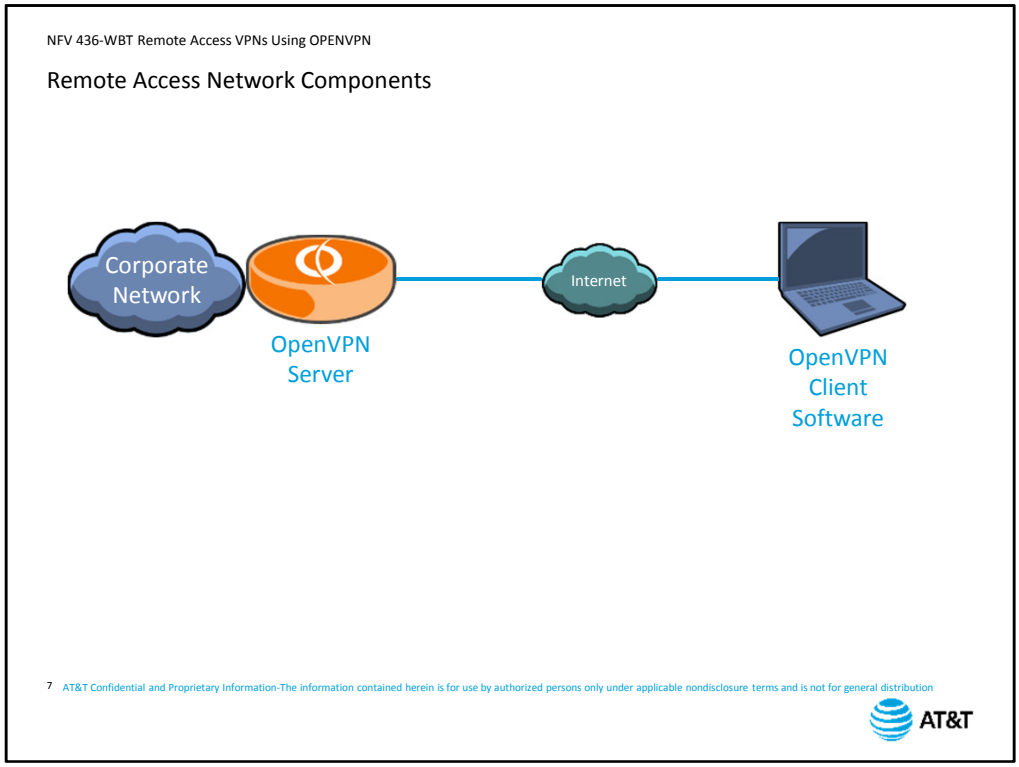
We'll begin with an overview of how OpenVPN provides secure remote access.



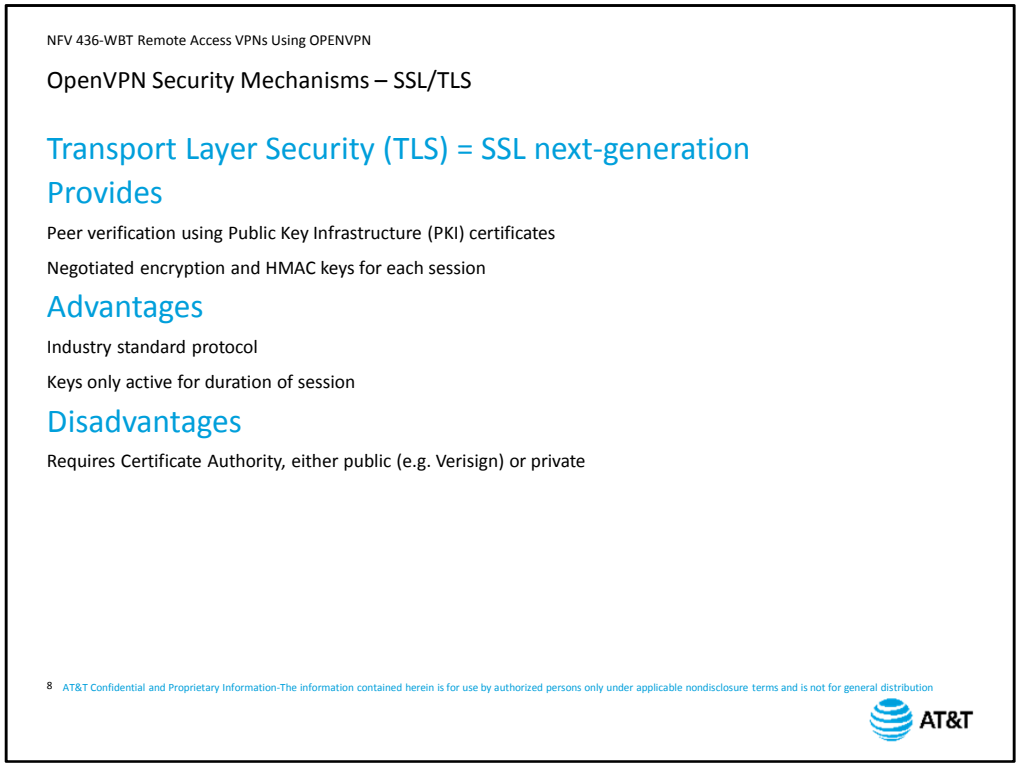
The vRouter implementation of remote access allows individual users to establish a secure connection to a private network using the Internet. Users first connect to the Internet using any service provider,
Then establish a secured, encrypted connection to the vRouter. The vRouter then provides access to services located inside the private network.
Once the connection is established, the remote user appears to be directly connected to the corporate network. All traffic from the remote computer travels to the private network before being routed to its destination,
including traffic destined for resources outside the private network.



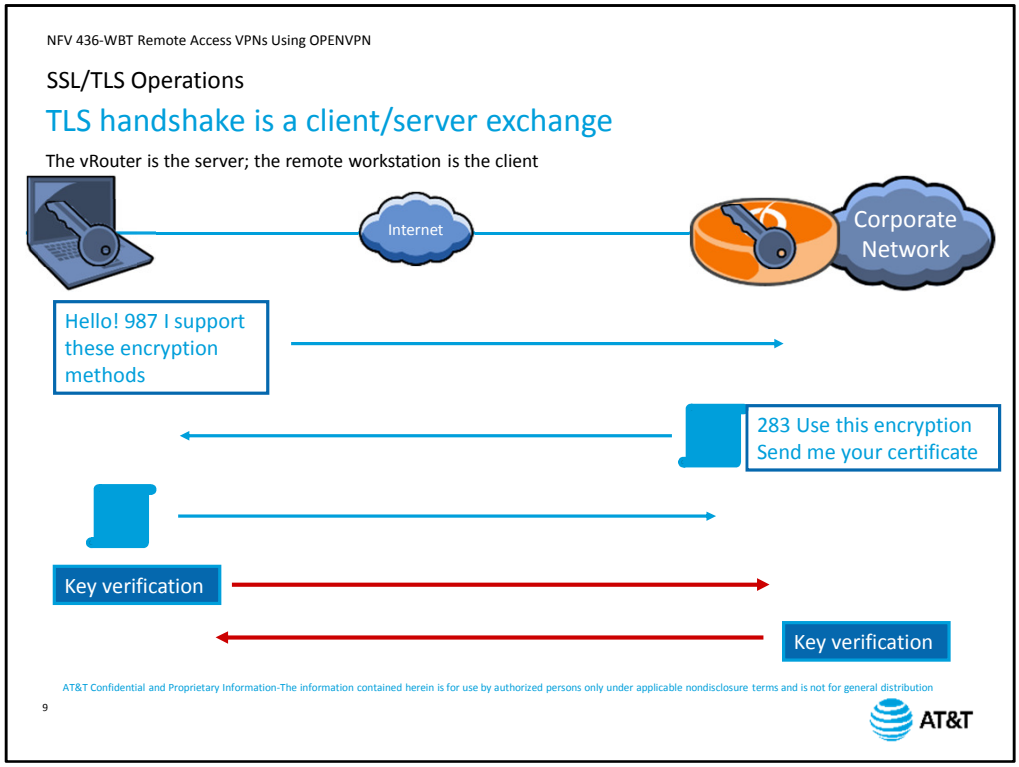
vRouter offers three options for remote access.
OpenVPN, which we cover in this course, is an open source solution for both site-to-site and remote-access secure communications.
OpenVPN provides client software for Windows, Linux, and Mac platforms. Note that the client software comes from OpenVPN directly and is not distributed by AT&T.
OpenVPN uses SSL/TLS to secure the connection.
The OpenVPN implementation is not compatible with other SSL-based remote access products, so you must use the OpenVPN client on the endpoints in order to enable remote access.
The other two remote access options are PPTP and L2TP. These are covered in detail in the AT&T vRouter Remote Access Using PPTP and L2TP course.



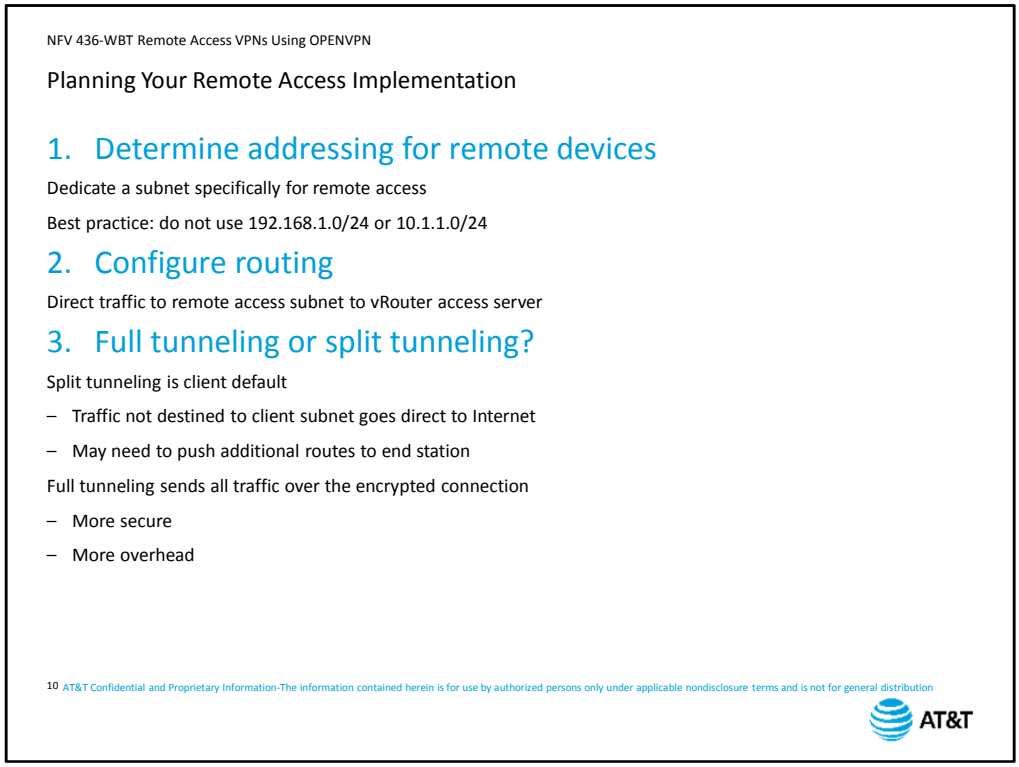
As we mentioned on the previous slide.
The end user workstation needs to run OpenVPN client software to initiate the connection, authenticate the end user, and encrypt the data exchanges.
The remote access server is the terminus for the secured connection, providing user authentication and data encryption services. The vRouter is a remote access server.



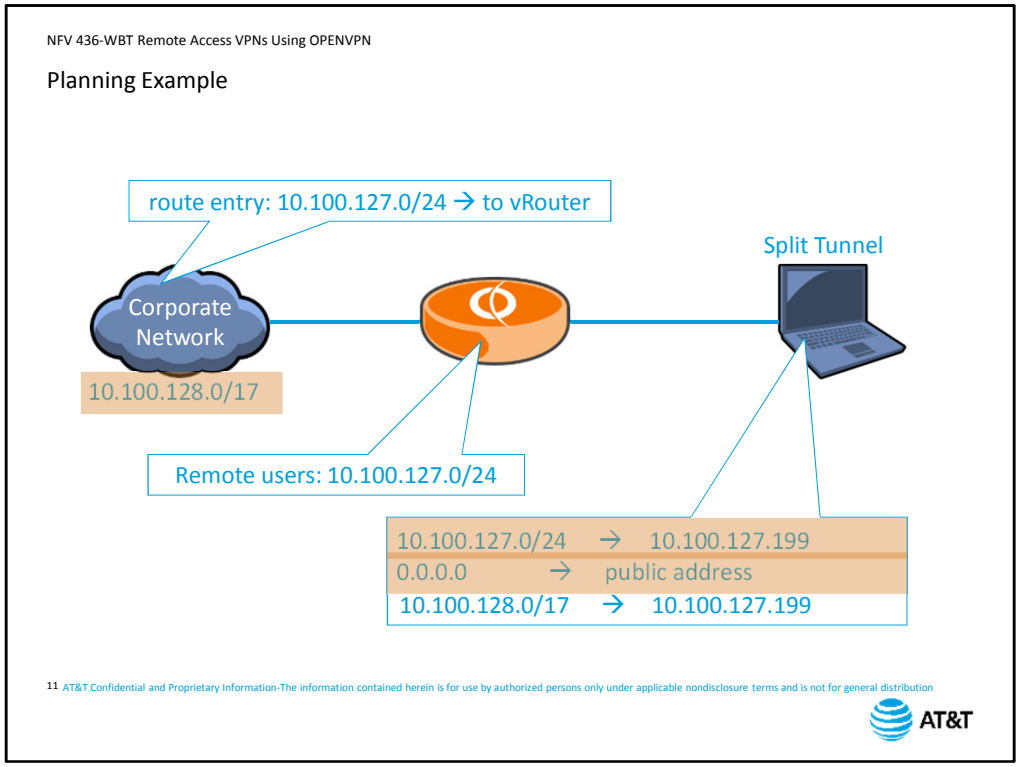
OpenVPN secures remote access using SSL/TLS.
Transport Layer Security (TLS), is simply the next-generation of Secure Socket Layer (SSL).
Originally developed to secure web communications, TLS is now widely used for securing all kinds of data exchanges.
TLS provides peer verification using the public key infrastructure (PKI). Devices exchange certificates, which are validated through the use of a Certificate Authority (CA). This validation takes place before any user data is exchanged.
TLS also provides dynamic key negotiation on a per-session basis.
TLS has a couple of advantages. TLS is an industry-standard protocol, so the peer validation and key exchange mechanisms are well-defined.
The negotiated keys are only active for the duration of the session.
The biggest disadvantage to using TLS is that it requires the use of a Certificate Authority to generate and sign device certificates. This can be a public CA, such as Verisign, or a privately-deployed certificate server, such as a Linux system running OpenSSL.



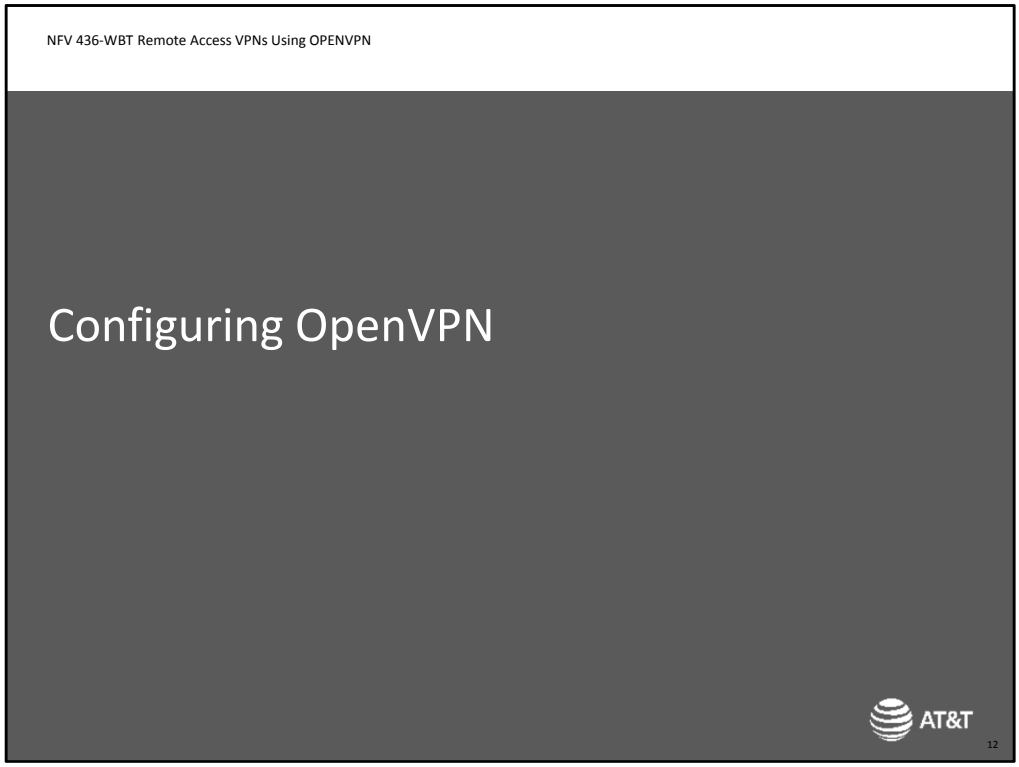
Because TLS was originally designed for secure Web-based communications, establishing a connection follows a client/server model.
The vRouter is the server, and the remote workstation is the client.
To begin establishing a connection,
the client sends a hello message to the server. This message contains a random number and the encryption and HMAC methods the client can support.
The server responds with its certificate, which allows the client to verify the identity of the server. The client uses the CA information to verify the server's certificate. If the server certificate cannot be verified, the client drops the connection. The server also sends a random number, the selected encryption and HMAC methods, and a request for the client's certificate.
If the server is validated, then the client sends his certificate to the server. The server uses the CA information to validate the client's certificate. Again, if the server cannot validate the client certificate, it will drop the connection. Verifying the certificates is where most connection attempts fail, usually because of missing or outdated files relating to certificate verification. We will discuss these required files in detail in the configuration section later in the course.
Assuming both certificates are validated, each device generates a session key using the random numbers provided in the first two messages.



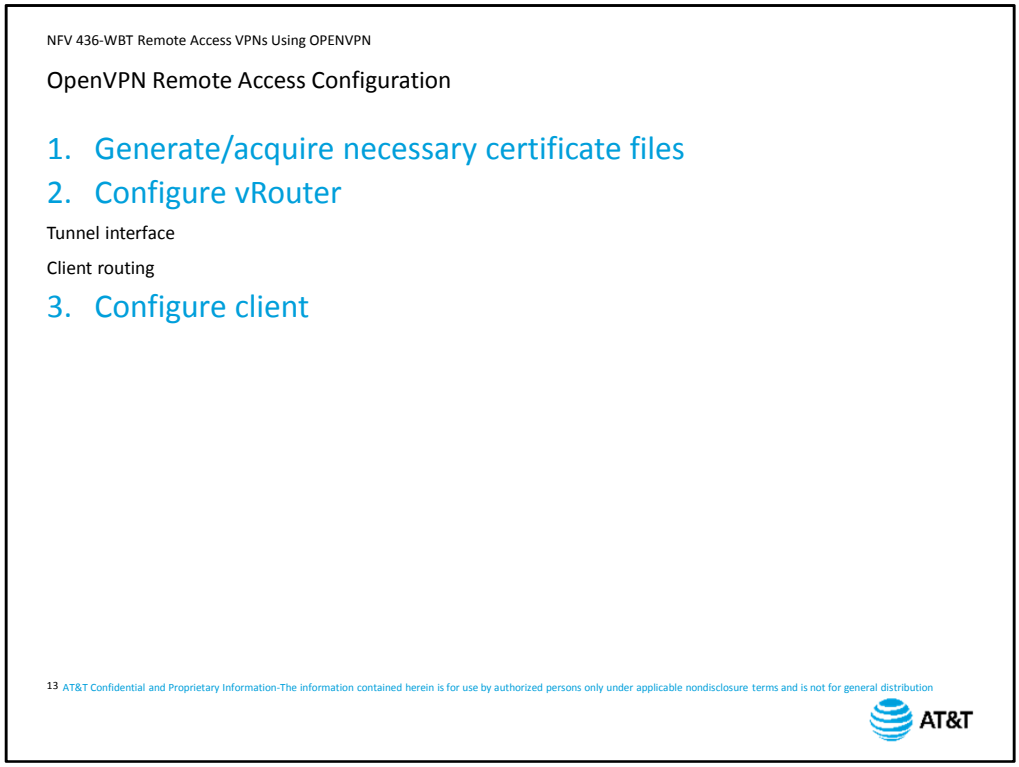
Before you begin configuring your vRouter for remote access, you need to do some planning and design.
First, you need to determine the range of addresses you will make available to remote users. This range of addresses should be a dedicated subnet of addresses and should not overlap with any subnets deployed within the private network.
In order to avoid conflicts with common default addressing schemes, you should also avoid using the subnets 192.168.1.0 or 10.1.1.0 for remote access devices.
Next, you need to configure routing within your network.
The subnet you dedicate for remote access is reachable via the vRouter. Upstream routers within the private network need to know to forward traffic for the remote access subnet to the vRouter. As this address range is not directly associated with a physical interface, you will need to configure static routing and route redistribution in order for upstream routers to learn about the remote access subnet.
Finally, you need to consider whether you want to use full tunneling or split tunneling.
Split tunneling is the default for OpenVPN clients. The only traffic to be tunneled by the OpenVPN client is traffic destined for hosts on the remote access subnet. All other traffic gets sent directly to the Internet without any encryption overhead.
You may need to add additional routes if you want to tunnel traffic to other subnets. We will look at an example on the next screen.



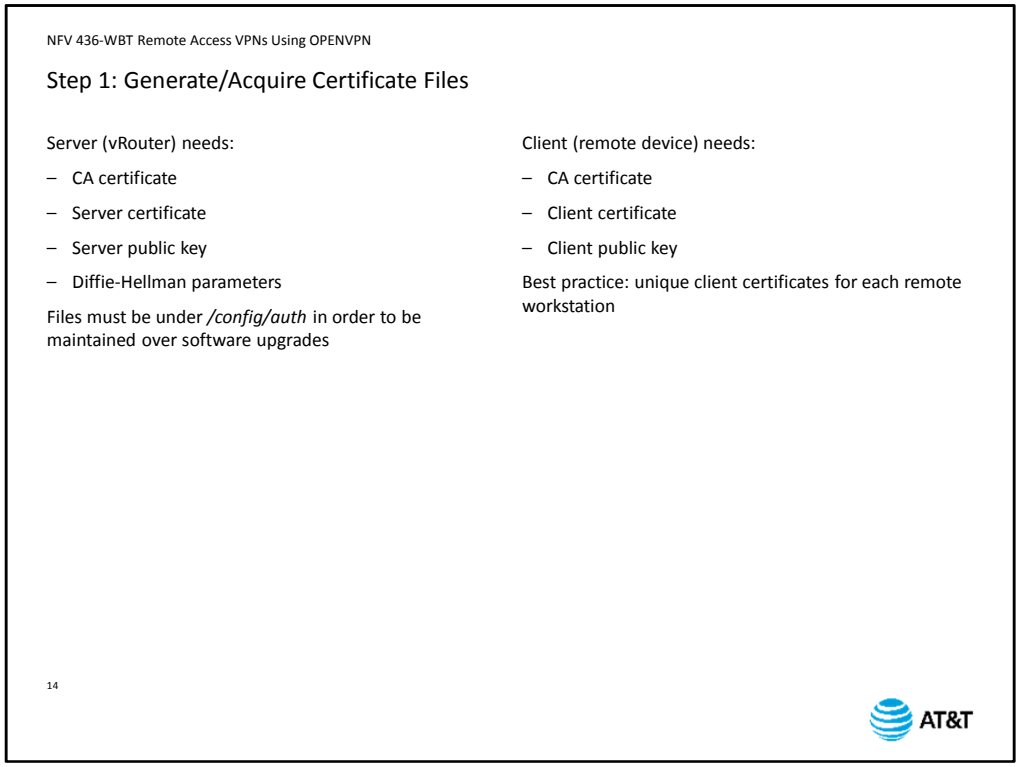
Let's look at a planning scenario.
We have decided to use subnet 10.100.127.0/24 for remote access users.
In order for devices within the corporate network to reach our remote users, we need to ensure that the routers know that hosts on subnet 10.100.127.0 are reachable through the vRouter.
When the client connects, the split tunnel configuration installs two routes.
The first route sends traffic to other hosts on the assigned subnet via the tunnel.
The other is a default route directing traffic directly to the Internet.
Note that if we want to reach hosts inside the private network, we need to add another route for the networks inside the corporate network. We can configure the vRouter to "push" this route to the client when it connects. This allows for central administration of those routes rather than having to directly configure them on the client.



Now we will look at the commands for configuring OpenVPN remote access.



The first step in setting up remote access VPN is to generate the certificate files needed by the server and the client, then copy those files to the devices. We will look at the list of files needed in detail on the next screen.
Next, you will configure the vRouter, setting up the tunnel interface and any routing needed to support client connectivity.
Finally, you will set up the client endpoint to connect to the vRouter.



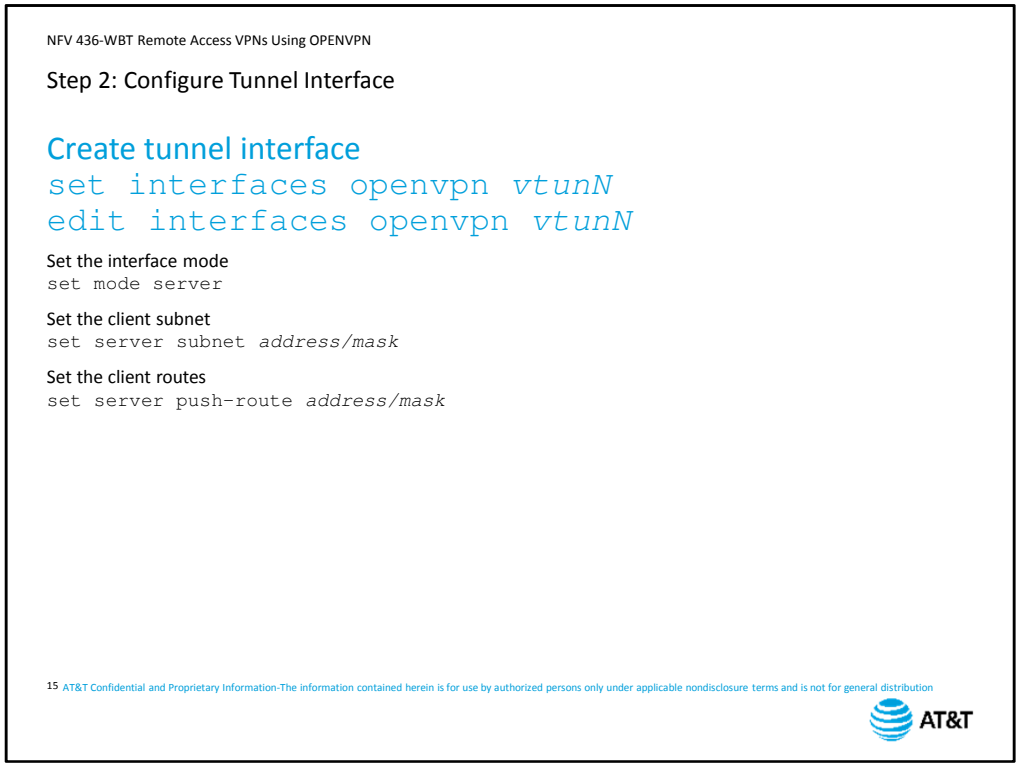
Your first step is to copy the necessary certificate files to each tunnel peer.
The server device needs:

- CA certificate
- Server certificate
- Server public key
- Diffie-Hellman parameters

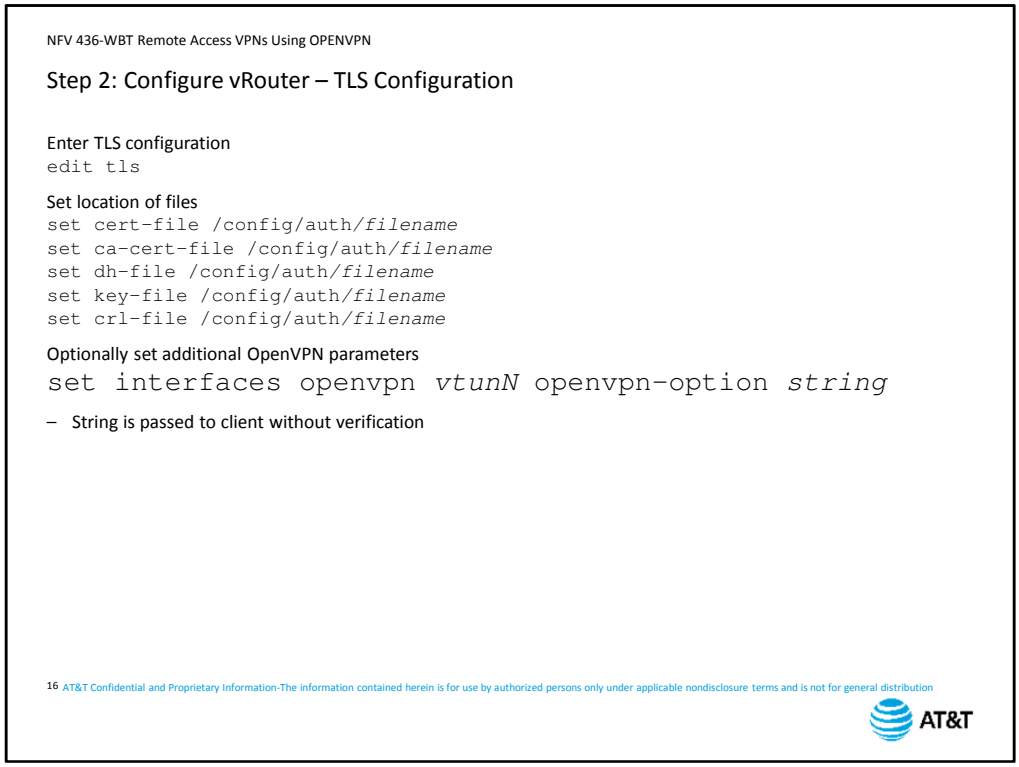
The files need to be under the */config/auth* directory structure. This will ensure that the authentication files are preserved across software upgrades. If the files are in a different directory, they may not persist over upgrades.
The client device needs:

- CA certificate
- Client certificate
- Client public key

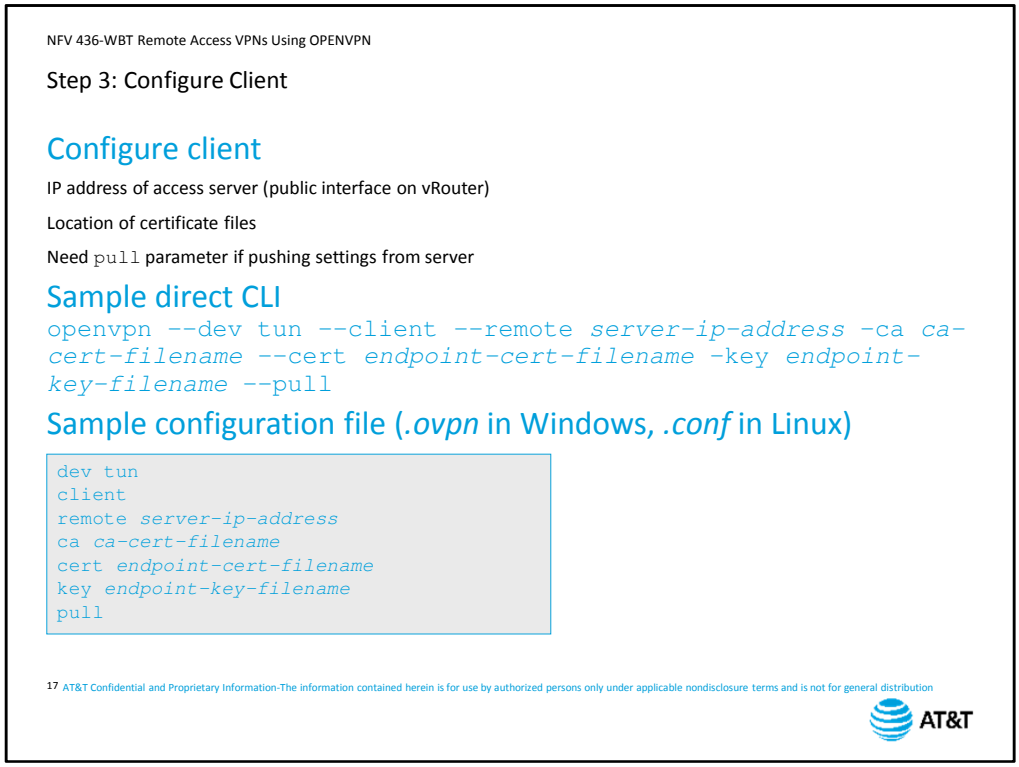
Although you can use the same set of client certificates on multiple remote workstations, the best practice is to generate a unique client certificate and public key for each remote workstation. How you acquire these files and the file name formats will vary depending on your certificate authority.



Once you have added the files to each device, configure the tunnel interface on the vRouter. First, create the tunnel interface using the command `set interfaces openvpn` followed by the tunnel number. The format is *vtun* followed by a number.
Because the rest of the commands must all be associated with the same tunnel interface, we recommend using the `edit` command to move you within the configuration hierarchy.
Set the interface mode to *server*. This instructs the vRouter to receive incoming SSL connection requests.
Next, set the subnet that will be used for client workstations.
Next, set the routing information for the client if needed. By default, the client will only tunnel traffic to hosts on the subnet assigned to clients. Since you most likely have other subnets within the private network that you want reachable via the tunnel, you will need to add these routes. You can add as many routes as you need for your configuration.



Next, use the `edit tls` command to move into the TLS level of the hierarchy.
Specify the location of all the files needed to perform certificate-based authentication. The Crl-file is optional, but all other files are required and must be present before you can commit your configuration.
OpenVPN itself has over 200 different options, and vRouter has not incorporated all of them into the CLI.
If you want to set additional OpenVPN parameters, vRouter has added a command that allows you to directly implement OpenVPN options.
The option string you enter is passed directly to the OpenVPN application and is not validated by the CLI. Therefore, you need to make sure that the specified option and associated values are valid, and there are no conflicts with any other OpenVPN options configured either through vRouter CLI commands or OpenVPN option commands.

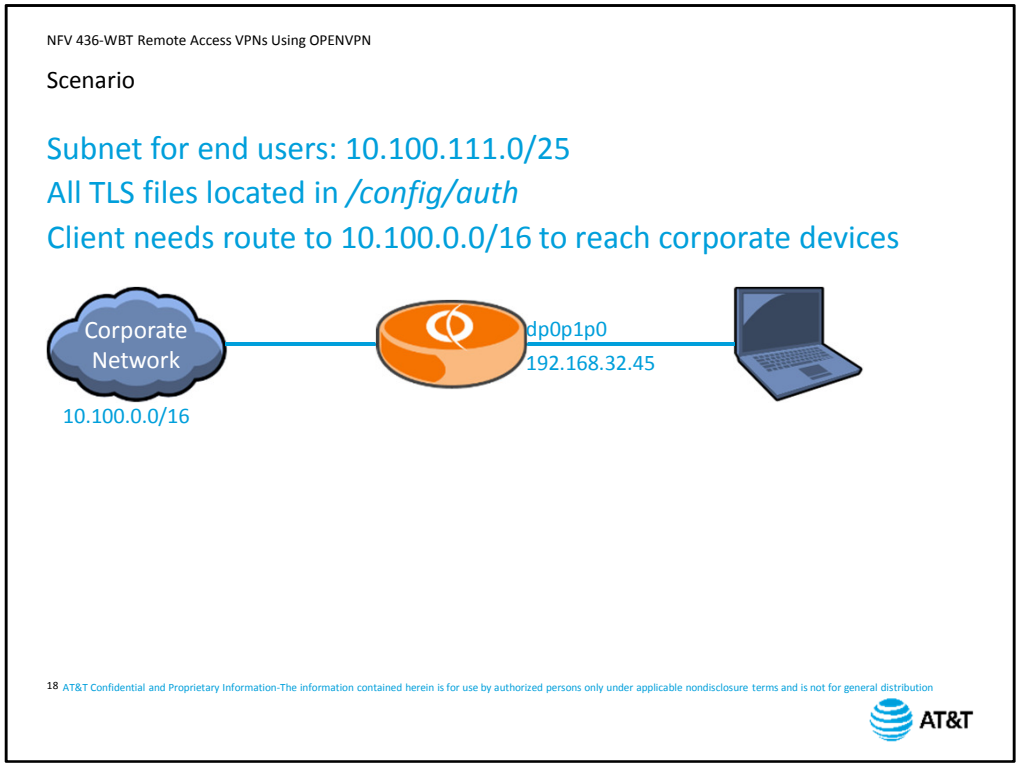


Next, you need to set up the client side of the connection. The client needs to know the IP address of the access server – in this case, the public address of the vRouter, and the location of the client certificate files.

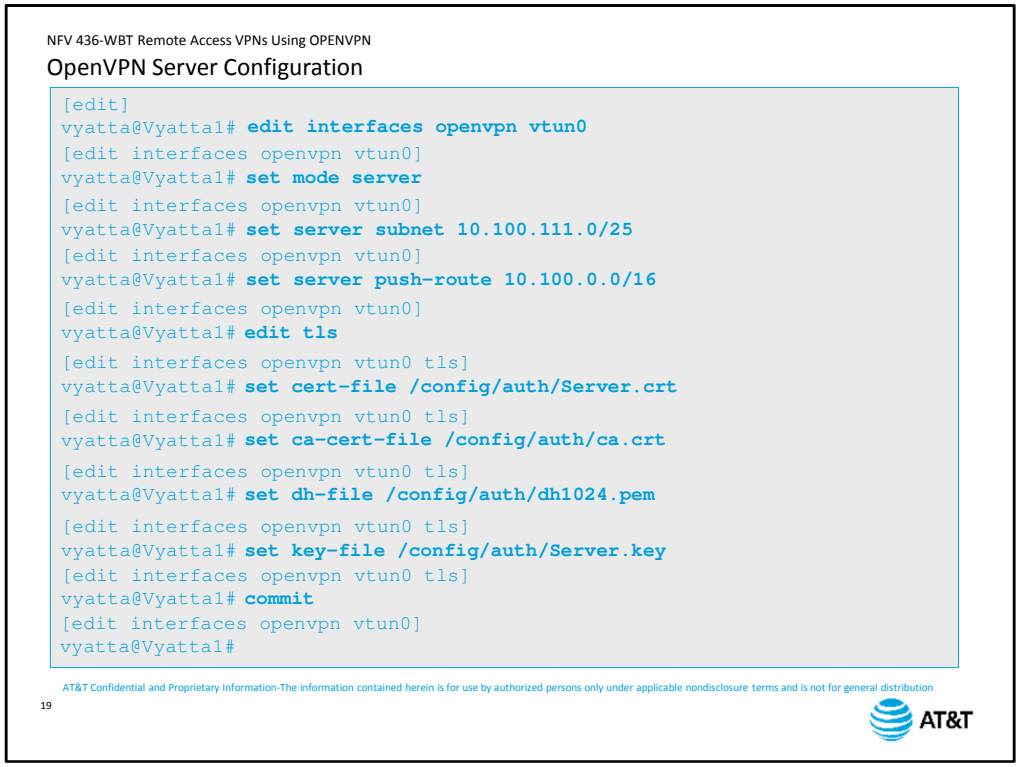
If you are pushing configuration information – such as routing – from the server, the client needs a corresponding `pull` setting in its configuration

You can pass these parameters directly to the client when you start up the software using a command line, or you can refer to a configuration file that loads the same parameters.

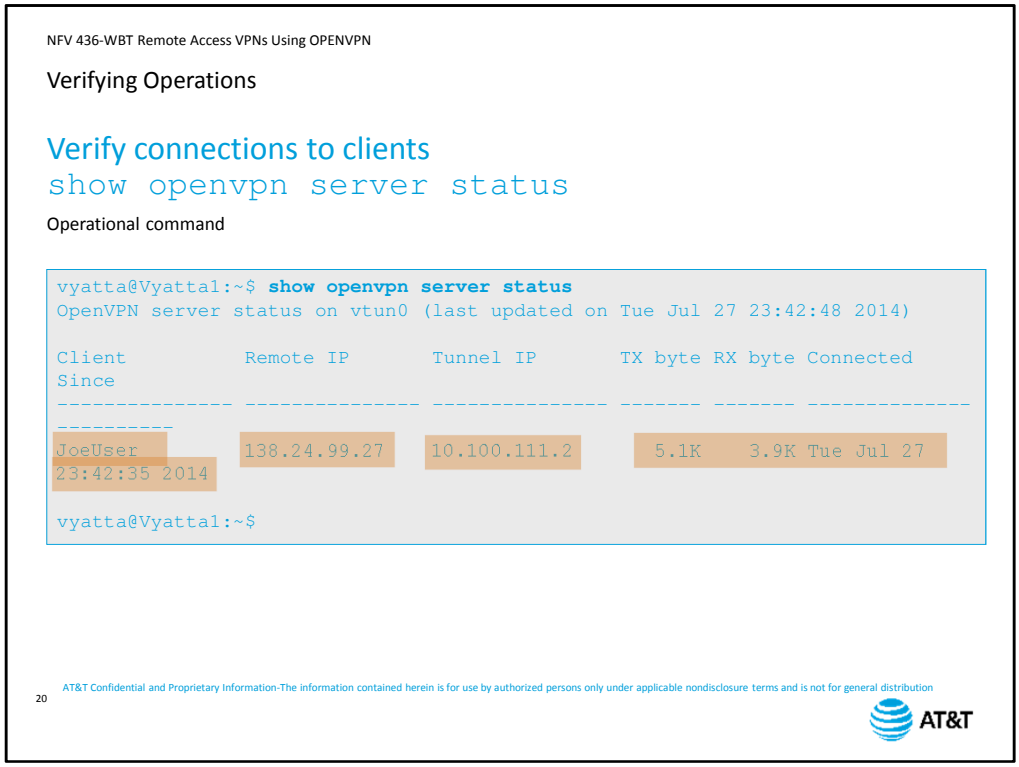
Specific client configuration differs for each platform, and is beyond the scope of this course.



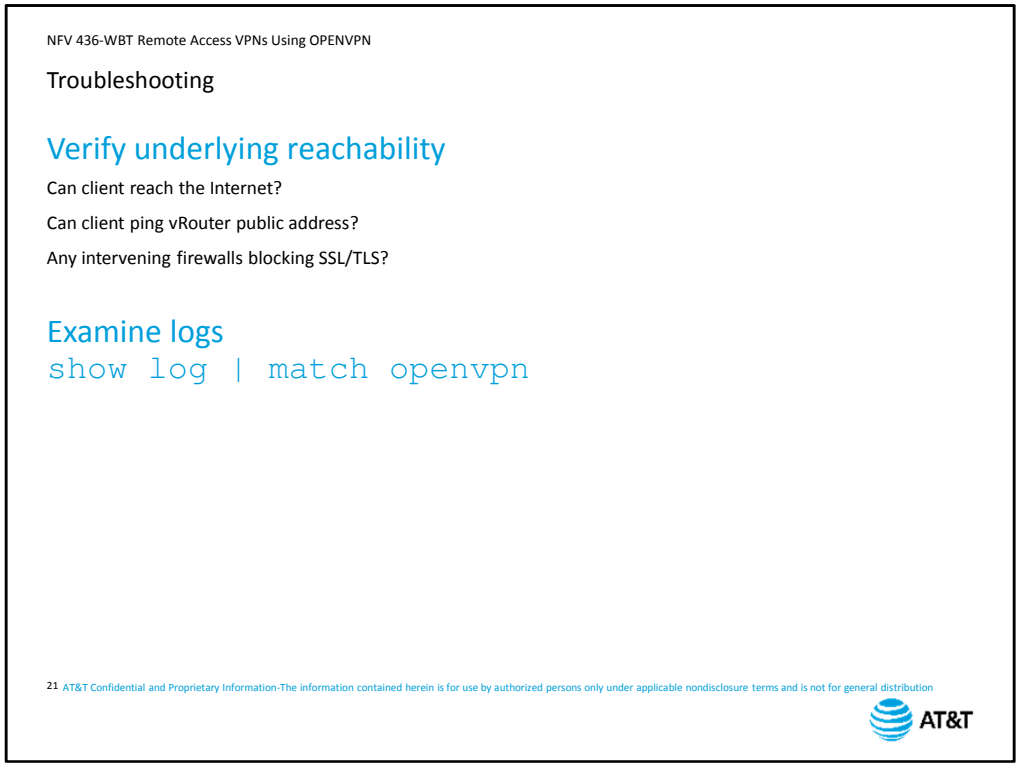
In this scenario, we have allocated a subnet for remote access – 10.100.111.0/25
We have already generated our certificate files and copied them to the */config/auth* directory of our device.
We also need to push a route to the client for subnet 10.100.0.0/16.



We begin the configuration by using the `edit` command to create the VPN tunnel interface and enter that level of the command hierarchy.
Next, we configure the OpenVPN mode to *server*. This instructs the vRouter to receive incoming TLS connection requests.
Next, we specify the range of addresses available to assign to remote users.
Next, we configure the route we want pushed to the client. This allows end users to reach the rest of the subnets inside our network over the tunnel.
Next we use the `edit` command to enter the TLS hierarchy
We specify the location of the certificate file,
the Certificate Authority certificate file,
the file containing the Diffie-hellman key generation parameters,
and the location of the servers' private key file.
We commit our changes to make them active. Not shown is the `save` command to make our changes permanent.



The best way to test your OpenVPN configuration is to connect your client. If the connection succeeds, you know your vRouter configuration is correct.
You can verify connectivity on the vRouter side with the Operational command `show openvpn server-status`. The output includes,
the name of the remote workstation as included in the client's certificate – in this case, the name is JoeUser,
the public IP address of the remote workstation,
the assigned tunnel address for the client (this address was taken from the subnet assigned to the interface),
and connection statistics.



When you are troubleshooting VPNs, it is important to first determine whether the problem is actually a VPN problem.

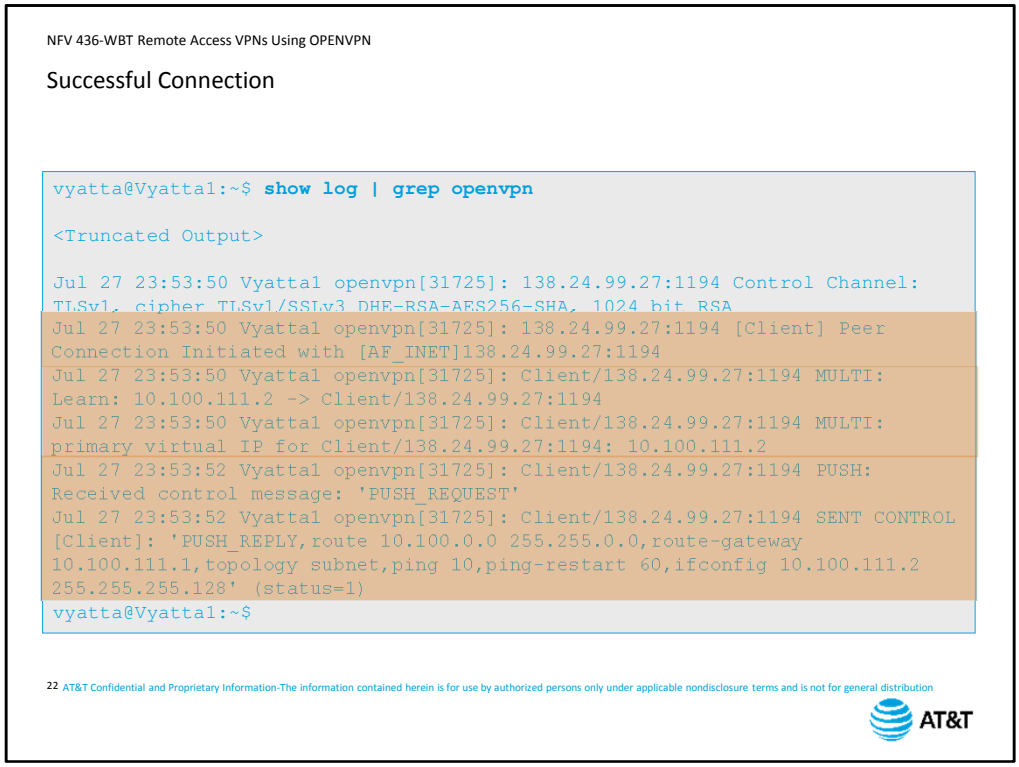
VPNs depend on the underlying network for connectivity. So, for example, if the end user is having a performance problem with his internet connection, their VPN is also going to have a performance problem.

Specifically, you need to verify that the client can reach the Internet itself. The VPN requires the internet in order to reach the public address of the vRouter.

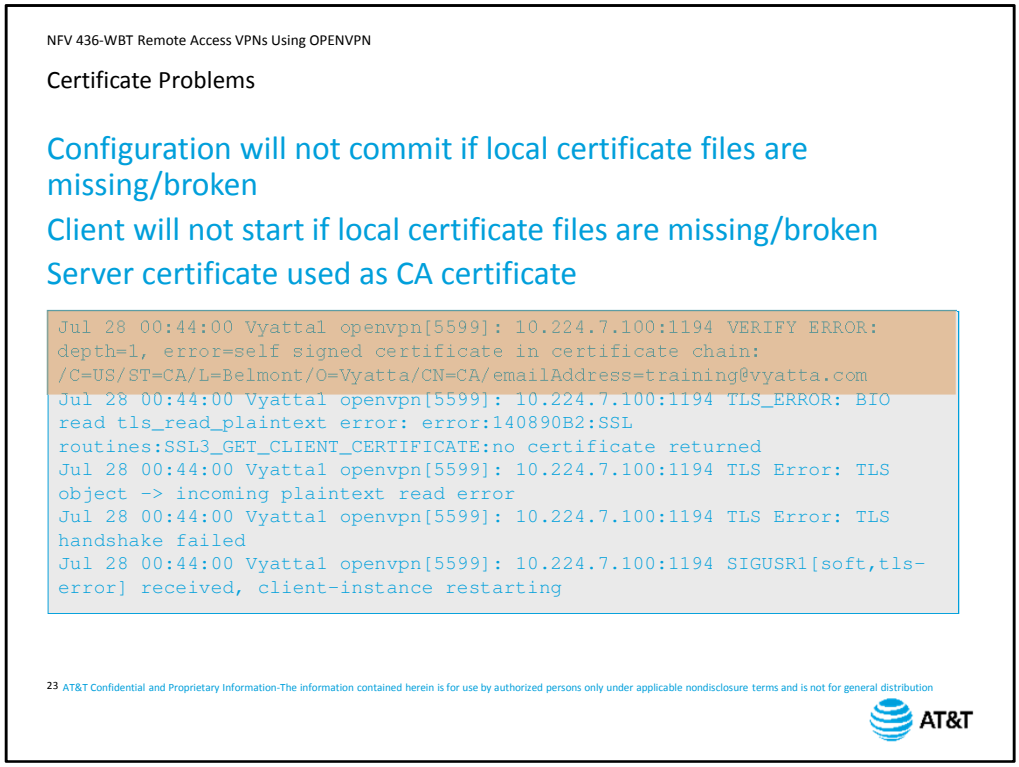
Next, verify that the client can reach the vRouter itself using ping or traceroute. Success here establishes that there's an IP path end to end between the client and the server.

Next, verify that there are no intervening firewalls that are blocking SSL/TLS traffic between the client and server.

If you have verified the underlying reachability, but the VPN still is not working, examine the logs on the vRouter, using the Linux pipe (|) capability and `grep` commands to limit the output to OpenVPN-related messages. You can use additional pipes and `grep` commands to further isolate the output to a specific IP address.



Let's look at the debug output for a successful connection.
You are looking for the message "Peer Connection initiated". This message indicates that the TLS negotiation was successful.
Next, the vRouter server assigns the client its tunnel IP address for the session.
Finally, you see the client requesting any pushed configuration data, and the vRouter returning the configured static route.

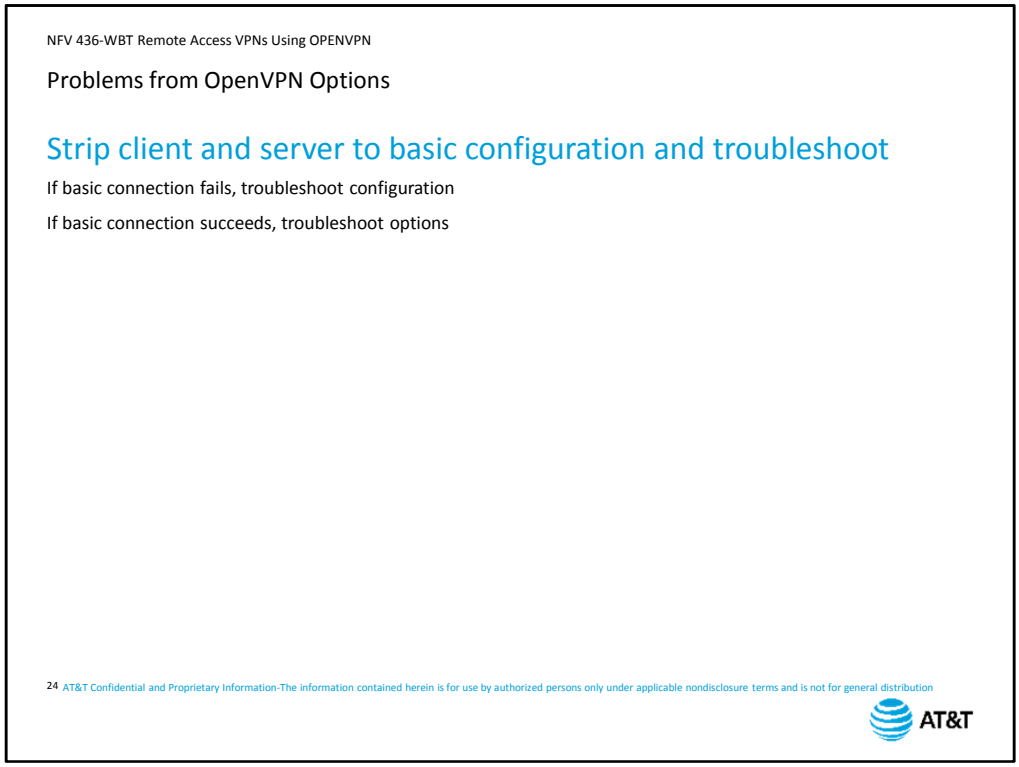


A common source of problems relates to the certificate files. Fortunately, many of the more common problems are easy to diagnose and repair.

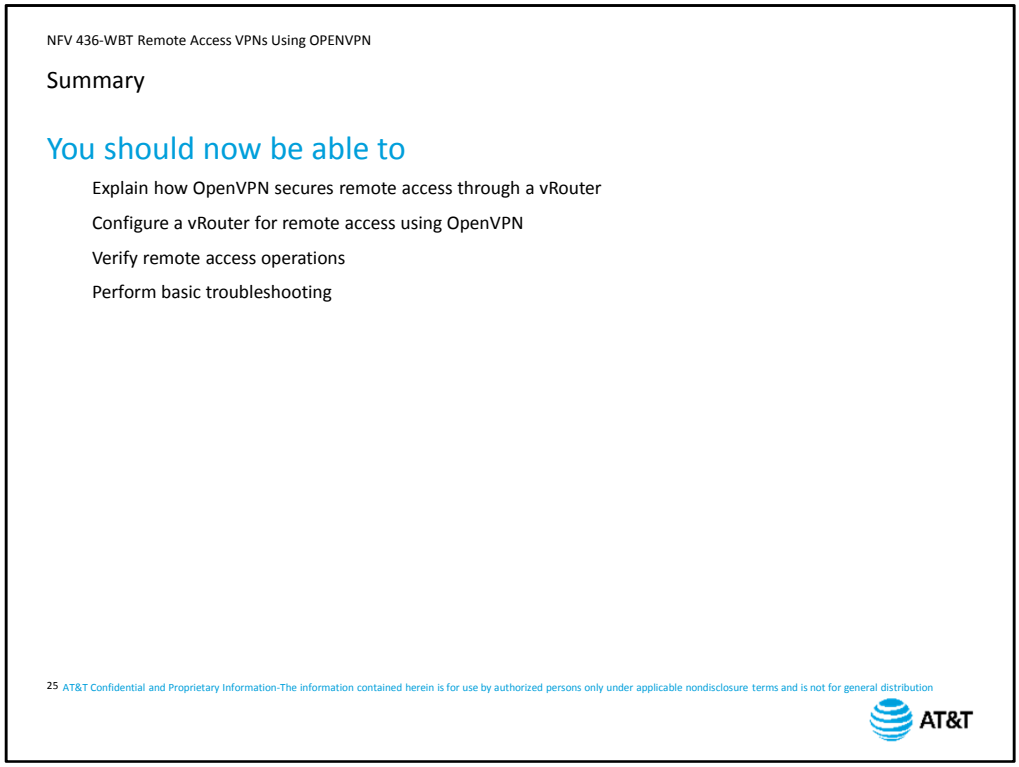
For example, the vRouter will not accept a configuration with missing or invalid certificate files. The `commit` command will fail and tell you which files are the source of the problem.

If the certificate files are missing or invalid on the client side, the client will not start. You will not see anything on the vRouter at all.

A possible problem on the vRouter side is that you mistakenly associate the server certificate with the ca-cert-file command. In this case, the debug output indicates a verification error when invoking the CA certificate.



Another possible source of problems is the overall complexity of OpenVPN and all the options available.
If you have got a complicated setup and are experiencing issues, strip your configuration back to the basic setup presented in this course.
If you are still experiencing issues, then you need to troubleshoot the vRouter configuration.
If not, then the issue relates to the options you have selected. Refer to the OpenVPN documentation for assistance.



Congratulations! You have completed the AT&T vRouter Remote Access Using OpenVPN course.

You should now be able to:

- Explain how OpenVPN secures remote access through a vRouter
- Configure the vRouter for remote access using OpenVPN
- Verify remote access operations
- Perform basic troubleshooting

We hope that this course has been useful, and that you will take additional AT&T University courses in the future.

End of Course – Remote Access VPNs Using OPENVPN