REPORT
OpenVPN Remote Access VPN Deployment and Validation

v1.0.0

Author:

Eldon Gabriel

January 20, 2026

TABLE OF CONTENTS

Disclaimer: This report documents independent work completed during a Mossé Cyber Security Institute (MCSI) lab exercise. It represents my personal understanding, configuration, and validation of an OpenVPN remote access VPN in a controlled lab environment. No MCSI video content, lab instructions, or proprietary materials are reproduced or disclosed. All content is presented in accordance with MCSI academic integrity and disclosure requirements.

REVISION HISTORY

Version	Date	Author	Description of Changes
v1.0.0	01/20/2026	Eldon G.	Initial draft.

1.0 OPENVPN REMOTE ACCESS VPN DEPLOYMENT

1.1 Project Description

This project involved deploying and validating an **OpenVPN** remote access **VPN** to securely tunnel client traffic from a Linux-based **VPN** server to the public Internet. The goal was to demonstrate a practical understanding of **VPN configuration**, **certificate-based authentication**, **routing**, **NAT**, **firewall rules**, and **end-to-end validation**.

The environment included the following factors:

- **OpenVPN** server running on **Ubuntu Linux**
- **Windows** client using **OpenVPN GUI**
- **UDP**-based **VPN** tunnel on `port 1194`
- Certificate and key-based authentication

This work was performed as an independent hands-on lab and documented for portfolio evidence.

1.2 VPN Server Configuration

The **OpenVPN** server was confirmed to be running via `systemd` using the instance-based service:

- **Service:** `openvpn-server@server`
- **Status:** `Active (running)`
- **Tunnel Network:** `10.8.0.0/24`
- **Tunnel Interface**: `tun`

The key server configuration elements included:

- `topology subnet`
- `push "redirect-gateway def1 bypass-dhcp"`
- `push "dhcp-option DNS 1.1.1.1"`
- `push "dhcp-option DNS 8.8.8.8"`

These settings ensured that the connected clients:

- Received an internal **VPN IP** address
- Routed all Internet traffic through the **VPN**
- Used known public **DNS** resolvers

1.3 Client Configuration and Certificate Handling

The Windows client configuration embedded all the required cryptographic materials directly into the `.ovpn` profile:

- CA certificate (`ca.crt`)
- Client certificate (`client1.crt`)
- Client private key (`client1.key`)
- TLS authentication key (`ta.key`)

Each file was opened in a text editor, and its full contents were pasted into the appropriate XML-style blocks:

CA certificate

```
<ca>

-----BEGIN CERTIFICATE-----
```

[REDACTED – Certificate Authority public certificate generated via **Easy-RSA**]. Used by the client to verify the authenticity of the OpenVPN server.]

```
-----END CERTIFICATE-----

</ca>
```

Client certificate

```
<cert>

-----BEGIN CERTIFICATE-----
```

[REDACTED – Client certificate issued by internal CA. Used by the **OpenVPN** server to authenticate the connecting client.]

```
-----END CERTIFICATE-----

</cert>
```

Client private key

```
<key>
-----BEGIN PRIVATE KEY-----
```

[REDACTED – client private key generated via **Easy-RSA**]

```
-----END PRIVATE KEY-----
</key>
```

TLS auth key

```
<tls-auth>
-----BEGIN OpenVPN Static key V1-----
```

[REDACTED – TLS authentication key (`ta.key`) generated on the server] Used to protect the control channel and mitigate unauthenticated packet attacks.]

```
-----END OpenVPN Static key V1-----
</tls-auth>
```

This eliminated the dependency on external key files and ensured the portability of the client configuration.

1.4 Networking, Routing, and Firewall Configuration

Initial connectivity issues were traced to missing packet forwarding and **NAT** rules.

The following checks and fixes were applied:

- **IPv4** forwarding was verified as being enabled:
  `net.ipv4.ip_forward = 1`
- `iptables` installed and verified
- **NAT** masquerading is configured to allow **VPN** client traffic to exit via the server's public interface

The firewall and cloud security group rules were configured to allow the following:

- **UDP** `1194` inbound from the client public **IP**

After applying these changes, the **VPN** tunnel was successfully established, and the traffic was routed correctly.

1.5 Troubleshooting and Resolution Summary

Key issues encountered and resolved:

- Connection timeouts caused by missing inbound **UDP** rules
- No Internet access caused by missing **NAT** rules
- The client was unable to validate the routing until the redirect-gateway was confirmed
- **ICMP** timeouts during `traceroute` are caused by intermediate routers filtering probes

Each issue was resolved through systematic verification of the following:

- Service status
- Routing tables
- Firewall rules
- Client-side testing

2.0 VALIDATION AND RESULTS

2.1 Internal VPN Connectivity Validation

The client successfully received an internal **VPN IP** address in the `10.8.0.0/24` range.

Validation command:

- `ping 10.8.0.1`

Result:

- 0% packet loss
- Stable latency

This confirmed the successful tunnel establishment and server reachability.

2.2 Internet Routing via VPN Validation

`Traceroute` was used to confirm that the Internet traffic traversed the **VPN** tunnel.

Validation command:

- `tracert 8.8.8.8`

Observed behavior:

- **First hop:** `10.8.0.1` (**VPN** gateway)
- **Subsequent hops:** Public Internet routers
- **Final hop:** `dns.google (8.8.8.8)`

This demonstrated that:

- The default route was pushed to the client
- Traffic exited through the **VPN** server
- **NAT** and forwarding were functioning correctly

3.0 CONCLUSION

3.1 Key Takeaways

- **OpenVPN** remote access **VPN** was successfully deployed and validated
- Certificate-based authentication was correctly implemented
- Routing and **NAT** are critical for client Internet access
- **ICMP** timeouts during traceroute do not necessarily indicate failure
- Systematic troubleshooting is essential for network services

---

3.2 Security Implications and Recommendations

- Restrict **VPN** access by source **IP** where possible
- Protect private keys and certificates with strong permissions
- Enable logging for authentication and tunnel events
- Regularly audit firewall and **NAT** rules
- Consider **MFA** integration for production deployments