Implementing a Custom VPN Using WireGuard and Routing Tables

WireGuard is a modern, fast, and simple VPN solution that's well-suited for both embedded devices and high-throughput servers. It uses state-of-the-art cryptography and operates at Layer 3 (network layer). This write-up covers how to implement a custom VPN using WireGuard and modify routing tables for secure communication between two endpoints.

2. WireGuard Basics

- **Interface**: `wg0`, `wg1`, etc.
- **Peers**: Defined by public key and allowed IPs.
- **Port**: Defaults to UDP 51820.
- **Configuration**: Can be done via `wg-quick` or directly with `wg`.

4. Configuration Files

Server `/etc/wireguard/wg0.conf`:

```
[Interface]
Address = 10.0.0.1/24
PrivateKey = <server-private-key>
ListenPort = 51820

[Peer]
PublicKey = <client-public-key>
AllowedIPs = 10.0.0.2/32
```

Client `/etc/wireguard/wg0.conf`:

```
[Interface]
Address = 10.0.0.2/24
PrivateKey = <client-private-key>
```

```
[Peer]
PublicKey = <server-public-key>
Endpoint = <server-ip>:51820
AllowedIPs = 0.0.0.0/0
PersistentKeepalive = 25
```

6. Starting WireGuard

```
sudo wg-quick up wg0
```

To stop:

```
sudo wg-quick down wg0
```

8. Routing Table Adjustments

You may want to route only specific subnets (split tunneling) or all traffic.

For split tunneling on client:

```
AllowedIPs = 10.0.0.0/24, 192.168.1.0/24
```

For full tunneling:

```
AllowedIPs = 0.0.0.0/0, ::/0
```

Check routes:

```
ip route show
```

10. Use Cases

- Secure remote access to private networks
- Site-to-site connectivity (e.g., branch office to HQ)
- VPN for Kubernetes pods or microservices

Summary

Implementing a VPN with WireGuard involves minimal setup but offers robust and performant security. Combined with correct routing table and firewall adjustments, it serves as a lightweight yet production-grade VPN solution suitable for many networking needs.