19: WireGuard - VPN Infrastructure

WireGuard is a modern VPN protocol using ChaCha20 encryption and Curve25519 key exchange. This installation uses WG-Easy (WireGuard-UI) for web-based client management, providing a user-friendly interface for VPN administration while maintaining WireGuard's performance and security benefits.

For protocol specifications, advanced configuration, and technical documentation, see the official WireGuard documentation.

Prerequisites

- **Docker installed** (Chapter 3)
- **Docker Compose** (Chapter 3)
- **Linux kernel 5.6+** - WireGuard kernel module support
- **Optional: Traefik installed** (Chapter 4) for HTTPS with Let's Encrypt
- **Optional: Domain configured** (Chapter 4.5), e.g., `vpn.example.com`
- **Firewall access** - Ability to open UDP port

Installation via Infinity Tools

Menu Installation

```
APPLICATIONS → WireGuard → Install
```

CLI Installation

```
sudo bash /opt/InfinityTools/Solutions/setup-wireguard.sh --install


# With domain (Traefik mode)
export WG_DOMAIN="vpn.example.com"
export WG_USE_TRAEFIK="true"
```

```
sudo -E bash /opt/InfinityTools/Solutions/setup-wireguard.sh --install

# Custom networks
export VPN_NETWORK_BASE="192.168.100"
export HOST_NETWORK_BASE="192.168.101"
export WG_VPN_PORT="51820"
sudo -E bash /opt/InfinityTools/Solutions/setup-wireguard.sh --install
```

Architecture

Containers

- **wireguard** - WG-Easy (WireGuard-UI) container (ngoduykhanh/wireguard-ui:latest)
- **wireguard-https** - Nginx SSL proxy (standalone HTTPS mode only)

Network Architecture

- **VPN Network:** Default `10.13.13.0/24` (configurable)
  - WireGuard server: `10.13.13.1`
  - Admin client: `10.13.13.2`
  - Client IPs: `10.13.13.3+` (auto-assigned)
- **Host Network:** Default `10.13.14.0/24` (configurable)
  - Host services IP: `10.13.14.1`
  - Accessible via VPN for host service access

Data Persistence

- **Data:** `/opt/speedbits/wireguard/data/` (WG-Easy database, client configs)
- **Configs:** `/opt/speedbits/wireguard/wg_confs/` (WireGuard config files)
- **SSL:** `/opt/speedbits/wireguard/ssl/` (standalone mode certificates)
- **Password:** `/opt/speedbits/wireguard/web-password.txt`

Host Integration

- **Kernel Module:** WireGuard kernel module loaded on host
- **Systemd Service:** `wireguard-host-network.service` for host network persistence
- **Network Interface:** `wg-host` dummy interface for host network
- **iptables Rules:** NAT and forwarding rules for VPN ↔ Host network

Deployment Modes

Traefik Mode

Uses Traefik for SSL termination and domain routing:

- Automatic Let's Encrypt certificate provisioning
- Domain-based access: `https://vpn.example.com`
- Security headers configured
- Requires: Traefik running, DNS A record configured

Standalone Mode (Default)

Direct access with HTTPS (self-signed):

- HTTPS: `https://SERVER_IP:8445` (self-signed cert via nginx proxy)
- Default web UI port: 8445 (configurable)
- VPN port: 51820 UDP (configurable)
- No domain required

Installation Process

Configuration Steps

1. **Network Configuration:** VPN network base (default: 10.13.13) and Host network base (default: 10.13.14)
2. **DNS Configuration:** Auto-detected from server's `/etc/resolv.conf`
3. **SSL Mode Selection:** Choose Traefik or Standalone
4. **VPN Port:** UDP port for VPN connections (default: 51820)
5. **Server Endpoint:** Public IP or domain name for client connections
6. **Kernel Module:** WireGuard kernel module installed and loaded
7. **Systemd Service:** Host network service created and enabled

What Gets Created

- **Directory:** `/opt/speedbits/wireguard`
- **Containers:** `wireguard`, `wireguard-https` (standalone mode)
- **Docker Compose:** `/opt/speedbits/wireguard/docker-compose.yml`
- **Systemd Service:** `/etc/systemd/system/wireguard-host-network.service`

- **Host Scripts:** `host-network-setup.sh`, `host-network-cleanup.sh`

Access Methods

Traefik Mode

```
https://vpn.example.com
```

Direct web access after DNS propagation and SSL certificate generation (30-60 seconds).

Standalone Mode

```
https://SERVER_IP:8445
```

Accept self-signed certificate warning (Advanced → Proceed).

Authentication

Web UI Credentials

- **Username:** `admin` (fixed)
- **Password:** Randomly generated (20 characters)
- **Storage:** `/opt/speedbits/wireguard/web-password.txt`
- **Hash:** bcrypt hash stored in container environment

VPN Client Authentication

- Each client gets unique public/private key pair
- Server validates client public key
- No username/password required for VPN connection
- Keys generated cryptographically secure

Network Configuration

VPN Network (10.13.13.0/24)

- **Purpose:** WireGuard clients and Docker services
- **Server IP:** `10.13.13.1`
- **Client IPs:** `10.13.13.3+` (auto-assigned)
- **Routing:** Clients can access Docker containers via container names

Host Network (10.13.14.0/24)

- **Purpose:** Access host services via VPN
- **Host IP:** `10.13.14.1`
- **Interface:** `wg-host` dummy interface
- **Routing:** NAT and forwarding rules configured

iptables Rules

```
# NAT for VPN → Host network
iptables -t nat -A POSTROUTING -s 10.13.13.0/24 -d 10.13.14.0/24 -j MASQUERADE

# Forwarding rules
iptables -A FORWARD -s 10.13.13.0/24 -d 10.13.14.0/24 -j ACCEPT
iptables -A FORWARD -s 10.13.14.0/24 -d 10.13.13.0/24 -j ACCEPT
```

Environment Variables

WireGuard Container

- `WGUI_USERNAME` - Web UI username (default: admin)
- `WGUI_PASSWORD` - Web UI password (randomly generated)
- `WGUI_SERVER_INTERFACE_ADDRESSES` - Server IP/CIDR (e.g., 10.13.13.1/24)
- `WGUI_SERVER_LISTEN_PORT` - VPN UDP port (default: 51820)
- `WGUI_DEFAULT_CLIENT_ALLOWED_IPS` - Default allowed IPs for clients
- `WGUI_DEFAULT_CLIENT_USE_SERVER_DNS` - Use server DNS (default: true)
- `WGUI_MANAGE_START` - Auto-start WireGuard (default: true)
- `WGUI_MANAGE_RESTART` - Auto-restart WireGuard (default: true)
- `SESSION_SECRET` - Session encryption key (randomly generated)

Client Management

Web UI Features

- Add/remove clients via web interface
- Generate QR codes for mobile devices
- Download .conf files for desktop clients
- Enable/disable clients individually
- View connection statistics
- Monitor traffic usage

Client Configuration

Clients are created via web UI. Each client gets:

- Unique public/private key pair
- Auto-assigned IP address
- Pre-configured AllowedIPs (VPN + Host networks)
- Server endpoint and public key

Security Configuration

Encryption

- **Cipher:** ChaCha20 (symmetric encryption)
- **Key Exchange:** Curve25519 (elliptic curve)
- **Hash:** BLAKE2s
- **Handshake:** Noise protocol framework

Access Security

- Traefik mode uses Let's Encrypt SSL (production-ready)
- Standalone HTTPS uses self-signed certificates (acceptable for internal use)
- Security headers configured (X-Frame-Options, CSP, etc.)
- Random password generation (20 characters)
- Unique keys per client

Container Security

- Requires `NET_ADMIN` and `SYS_MODULE` capabilities
- IP forwarding enabled
- Kernel module access for WireGuard
- Host network access for routing

Firewall Configuration

Required Ports

- **UDP 51820** (or custom VPN port) - VPN connections (MUST be open)
- **TCP 8445** (standalone mode) - Web UI (optional, can be closed after VPN setup)
- **TCP 443** (Traefik mode) - Web UI via Traefik

Firewall Best Practices

```
# Open VPN port (REQUIRED)
sudo ufw allow 51820/udp

# Close other public ports (access via VPN instead)
sudo ufw delete allow 8443  # Webmin
sudo ufw delete allow 8444  # Apprise
sudo ufw delete allow 8445  # WireGuard web UI
```

Systemd Service

Host Network Service

Service: `wireguard-host-network.service`

- Creates `wg-host` dummy interface
- Configures host network IP (10.13.14.1)
- Sets up routing and iptables rules
- Persists across reboots

Service Management

```
# Check status
systemctl status wireguard-host-network.service

# Restart service
sudo systemctl restart wireguard-host-network.service
```

```
# View logs
journalctl -u wireguard-host-network.service
```

Troubleshooting

VPN Connection Issues

- Verify firewall: `sudo ufw status | grep 51820`
- Check server endpoint accessibility
- Verify client config (AllowedIPs, endpoint, keys)
- Check WireGuard logs: `docker logs wireguard`
- Test UDP connectivity: `nc -u -v SERVER_IP 51820`

Host Network Issues

- Check systemd service: `systemctl status wireguard-host-network.service`
- Verify interface: `ip addr show wg-host`
- Check routing: `ip route show | grep 10.13.14`
- Verify iptables rules: `iptables -t nat -L -n -v`

Web UI Issues

- Check container status: `docker ps | grep wireguard`
- View logs: `docker logs wireguard`
- Verify password: `cat /opt/speedbits/wireguard/web-password.txt`
- Test API: `curl -u admin:PASSWORD http://localhost:5000/api/sessions`

Production Considerations

- **Access Method:** Use Traefik mode for production (trusted SSL)
- **Firewall:** Open only VPN port, close other public ports
- **Password Management:** Store web UI password securely
- **Client Management:** Regularly review and disable unused clients
- **Monitoring:** Monitor VPN connections and traffic
- **Backup:** Backup client configurations and keys
- **Updates:** Re-run install script periodically for updates

Integration with Infinity Tools

WireGuard complements Infinity Tools by providing:

- Secure remote access to all Infinity Tools applications
- Access to Docker services without exposing ports publicly
- Access to host services (Webmin, Apprise) securely
- Centralized VPN management via web interface

Recommended Setup:

- Open only VPN port (UDP 51820) publicly
- Close other public ports (Webmin, Apprise, etc.)
- Access all services via VPN
- Use VPN network (10.13.13.x) for Docker services
- Use Host network (10.13.14.1) for host services

Advanced Configuration

Custom Networks

Configure custom network ranges:

```
export VPN_NETWORK_BASE="192.168.100"
export HOST_NETWORK_BASE="192.168.101"
sudo -E bash setup-wireguard.sh --install
```

Custom VPN Port

```
export WG_VPN_PORT="51821"
sudo -E bash setup-wireguard.sh --install
```

Custom DNS

```
export VPN_DNS="8.8.8.8,8.8.4.4"
sudo -E bash setup-wireguard.sh --install
```

Client Configuration Export

Via Web UI

- Download .conf file for desktop clients
- Scan QR code for mobile clients
- View client details (IP, public key, etc.)

Via Command Line

```
# View admin client config
sudo bash setup-wireguard.sh --show-config

# Client configs stored in
ls /opt/speedbits/wireguard/data/peer_*/peer.conf
```

Next Steps

WireGuard is now operational. Use it to:

- Create VPN clients for all your devices
- Access Infinity Tools applications securely
- Access host services without exposing ports
- Manage clients via web interface
- Monitor VPN connections and traffic

For advanced features, protocol specifications, and development guides, refer to the official WireGuard documentation.

Revision #3
Created 17 November 2025 17:17:51 by bjoern
Updated 17 November 2025 17:22:16 by bjoern