WireGuard

APPLICATION NOTE

Used symbols

Danger – Information regarding user safety or potential damage to the router.

Attention – Problems that can arise in specific situations.

Information, notice – Useful tips or information of special interest.

Example – Example of function, command or script.

Open Source Software License

The software in this device uses various pieces of open source software governed by following licenses: GPL versions 2 and 3, LGPL version 2, BSD-style licenses, MIT-style licenses. The list of components, together with complete license texts, can be found on the device itself: See the *Licenses* link at the bottom of the router's main Web page (*General Status*) or point your browser to address DEVICE_IP/licenses.cgi. If you are interested in obtaining the source, please get in touch with us at:

techSupport@advantech-bb.com

Modifications and debugging of LGPL-linked executables:

The device manufacturer with this grants the right to use debugging techniques (e.g., decompilation) and make customer modifications of any executable linked with a LGPL library for its purposes. Note these rights are limited to the customer's usage. No further distribution of such modified executables and no transmission of the information obtained during these actions may be done.

Advantech Czech s.r.o., Sokolska 71, 562 04 Usti nad Orlici, Czech Republic.
Document No. APP-0103-EN, revision from September 2, 2022. Released in the Czech Republic.

Contents

List of Figures

List of Tables

1. WireGuard protocol

WireGuard is a secure network tunnel, operating at layer 3, implemented as a kernel virtual network interface for Linux, which aims to replace both IPsec for most use cases, as well as popular user space and/or TLS-based solutions like OpenVPN, while being more secure, more performant, and easier to use. The virtual tunnel interface is based on a proposed fundamental principle of secure tunnels: an association between a peer public key and a tunnel source IP address. It uses a single round trip key exchange, based on NoiseIK, and handles all session creation transparently to the user using a novel timer state machine mechanism. Short pre-shared static keys—Curve25519 points—are used for mutual authentication in the style of OpenSSH. The protocol provides strong perfect forward secrecy in addition to a high degree of identity hiding. Transport speed is accomplished using ChaCha20Poly1305 authenticated-encryption for encapsulation of packets in UDP. An improved take on IP-binding cookies is used for mitigating denial of service attacks, improving greatly on IKEv2 and DTLS's cookie mechanisms to add encryption and authentication. The overall design allows for allocating no resources in response to received packets, and from a systems perspective, there are multiple interesting Linux implementation techniques for queues and parallelism.

1.1 How does WireGuard work?

WireGuard works by adding a network interface (or multiple), like eth0 or wlan0, called wg0 (or wg1, wg2, wg3, etc). Routes can be installed on this network interface. The specific WireGuard aspects of the interface are configured using the wg(8) tool. This interface acts as a tunnel interface.

WireGuard associates tunnel IP addresses with public keys and remote endpoints. When the interface sends a packet to a peer, it does the following:

1. This packet is meant for 192.168.30.8. Which peer is that? Let me look... Okay, it's for peer *ABCDEFGH*. (Or if it's not for any configured peer, drop the packet.)
2. Encrypt entire IP packet using peer *ABCDEFGH*'s public key.
3. What is the remote endpoint of peer *ABCDEFGH*? Let me look... Okay, the endpoint is UDP port 53133 on host 216.58.211.110.
4. Send encrypted bytes from step 2 over the Internet to 216.58.211.110:53133 using UDP.

When the interface receives a packet, this happens:

1. I just got a packet from UDP port 7361 on host 98.139.183.24. Let's decrypt it!
2. It decrypted and authenticated properly for peer *LMNOPQRS*. Okay, let's remember that peer *LMNOPQRS*'s most recent Internet endpoint is 98.139.183.24:7361 using UDP.

3. Once decrypted, the plain-text packet is from 192.168.43.89. Is peer *LMNOPQRS* allowed to be sending us packets as 192.168.43.89?
4. If so, accept the packet on the interface. If not, drop it.

At the heart of WireGuard is a concept called Cryptokey Routing, which works by associating public keys with a list of tunnel IP addresses that are allowed inside the tunnel. Each network interface has a private key and a list of peers. Each peer has a public key. Public keys are short and simple, and are used by peers to authenticate each other. They can be passed around for use in configuration files by any out-of-band method, similar to how one might send their SSH public key to a friend for access to a shell server.

For example, a server computer might have this configuration:

```
[Interface]
PrivateKey = yAnz5TF+lXXJte14tji3zlMNq+hd2rYUIgJBgB3fBmk=
ListenPort = 51820

[Peer]
PublicKey = xTIBA5rboUvnH4htodjb6e697QjLERt1NAB4mZqp8Dg=
AllowedIPs = 10.192.122.3/32, 10.192.124.1/24
```

And a client computer might have this simpler configuration:

```
[Interface]
PrivateKey = gI6EdUSYvn8ugXOt8QQD6Yc+JyiZxIhp3GInSWRfWGE=
ListenPort = 21841

[Peer]
PublicKey = HIgo9xNzJMWLKASShiTqIybxZOU3wGLiUeJ1PKf8ykw=
Endpoint = 192.95.5.69:51820
AllowedIPs = 0.0.0.0/0
```

In the server configuration, each peer (a client) will be able to send packets to the network interface with a source IP matching his corresponding list of allowed IPs. For example, when a packet is received by the server from peer gN65BkIK..., after being decrypted and authenticated, if its source IP is 10.10.10.230, then it's allowed onto the interface; otherwise it's dropped.

In the server configuration, when the network interface wants to send a packet to a peer (a client), it looks at that packet's destination IP and compares it to each peer's list of allowed IPs to see which peer to send it to. For example, if the network interface is asked to send a packet with a destination IP of 10.10.10.230, it will encrypt it using the public key of peer gN65BkIK..., and then send it to that peer's most recent Internet endpoint.

In the client configuration, its single peer (the server) will be able to send packets to the network interface with any source IP (since 0.0.0.0/0 is a wildcard). For example, when a packet is received from peer HIgo9xNz..., if it decrypts and authenticates correctly, with any source IP, then it's allowed onto the interface; otherwise it's dropped.

In the client configuration, when the network interface wants to send a packet to its single peer (the server), it will encrypt packets for the single peer with any destination IP address (since 0.0.0.0/0 is a wildcard). For example, if the network interface is asked to send a packet with any destination IP, it will encrypt it using the public key of the single peer HIgo9xNz..., and then send it to the single peer's most recent Internet endpoint.

In other words, when sending packets, the list of allowed IPs behaves as a sort of routing table, and when receiving packets, the list of allowed IPs behaves as a sort of access control list.

This is what we call a Cryptokey Routing Table: the simple association of public keys and allowed IPs.

Any combination of IPv4 and IPv6 can be used, for any of the fields. WireGuard is fully capable of encapsulating one inside the other if necessary.

Because all packets sent on the WireGuard interface are encrypted and authenticated, and because there is such a tight coupling between the identity of a peer and the allowed IP address of a peer, system administrators do not need complicated firewall extensions, such as in the case of IPsec, but rather they can simply match on "is it from this IP? on this interface?", and be assured that it is a secure and authentic packet. This greatly simplifies network management and access control, and provides a great deal more assurance that your iptables rules are actually doing what you intended for them to do.

1.2 Restrictions in Advantech routers

- Routers allow to create only four WireGuard tunnels simultaneously
- Routers can not be used as a multiclient server

2. Configuration of WireGuard tunnel

WireGuard is not supported in v1 and v2 router platforms. In v2i and v3 routers, the IPv4 and IPv6 tunnels are supported.

WireGuard tunnel allows protected connection of four networks LAN to the one network. To open the *WireGuard* tunnel configuration page, click *WireGuard* in the *Configuration* section of the main menu. The menu item will expand and you will see four separate configuration pages: *1st Tunnel*, *2nd Tunnel*, *3rd Tunnel* and *4th Tunnel*. Description of all items is listed in following table.

Item	Description
Create 1st|2nd|3rd|4th WireGuard tunnel	If enabled, the tunnel is activated.
Description	Description (or name) of tunnel.
Host IP Mode	Select IP Mode, the choices are • IPv4 • IPv6
Remote IP Address	IP address of the opposite tunnel side (domain name can be used).
Remote Port	Port of the opposite tunnel side
Listen Port	Port where WireGuard listens/accepts connections
NAT/Firewall Transversal	When this option is enabled, a keepalive packet is sent to the server endpoint once every 25 seconds. If you don't need this feature, don't enable it. But if you're behind NAT or a firewall and you want to receive incoming connections long after network traffic has gone silent, this option will keep the "connection" open in the eyes of NAT.
Interface IPv4/IPv6 Address	Defines a IP address for assigning to wgX interface. WireGuard has 4 interfaces wg1 to wg4 according to a tunnel order, interface wgX has to have IPv4 or IPv6 address or both.
Interface IPv4/IPv6 Prefix	Defines the IPv4/IPv6 prefix of the interface of opposite tunnel side.

Continued on next page

Continued from previous page

Item	Description
Install Routes	If Yes is selected remote (local) subnets are used as traffic selectors (routes). If No is selected routes are not installed, subnet packet (that belong to the set Subnet) processing is enabled in kernel but routing is processed in another way, eg by routing protocol (Router App FRR / BGP or FRR / staticd)
Traffic Selector	• All traffic – All traffic goes to tunnel route 0.0.0.0/0, ::/0 • Subnets – Name only certain networks
Pre-shared Key	Optional choice for enhancing security.
Local Private Key & Local Public Key	Local key pair
Remote Public Key	Public key of the opposite side

Table 1: Configuration of OpenVPN tunnel

The changes in settings will be applied after pressing the *Apply* button.

1st WireGuard Tunnel Configuration

☐ Create 1st WireGuard Tunnel

Description *	
Host IP Mode *	IPv4
Remote IP Address *	
Remote Port *	
Listen Port	51820
NAT/Firewall Tranversal	no
Interface IPv4 Address *	
Interface IPv4 Prefix *	
Interface IPv6 Address *	
Interface IPv6 Prefix *	
Install Routes	no
Traffic Selector	Subnets
Subnets *	
Pre-shared Key *	Generate
Local Private Key	Generate
Local Public Key *	
Remote Public Key	

** can be blank*

Apply

Figure 1: Configuration form for WireGuard tunnel

3. Configuration example

3.1 Advantech router on both sides of the WireGuard tunnel

Configuration of the first router:

Item	Value
Listen Port	51820
NAT/Firewall Traversal	no
Interface IPv4 Address	10.0.0.1
Interface IPv4 Prefix	30
Instal Routes	Yes
Traffic Selector	Subnets
Subnets	172.16.24.0/24
Local Public Key	uSa2f1uyYq6z1uOd4Y9jeSHq5PiYpUneWDcBJdtqyis=
Remote Public Key	D2OYEa0MSQR0ONI8CBaqUJJQvXo4CSqmXdE+/3x+Zxc=

Table 2: Configuration of the first router

Configuration of the second router:

Item	Value
Remote IP Address	10.80.0.27
Remote Port	51820
Listen Port	51820
NAT/Firewall Traversal	yes
Interface IPv4 Address	10.0.0.2
Interface IPv4 Prefix	30
Install Routes	yes
Traffic Selector	Subnets
Subnets	172.16.24.0/24
Local Public Key	D2OYEa0MSQR0ONI8CBaqUJJQvXo4CSqmXdE+/3x+Zxc=
Remote Public Key	uSa2f1uyYq6z1uOd4Y9jeSHq5PiYpUneWDcBJdtqyis=

Table 3: Configuration of the second router

1st WireGuard Tunnel Configuration

☑ Create 1st WireGuard Tunnel	
Description *	
Host IP Mode *	IPv4
Remote IP Address *	
Remote Port *	
Listen Port	51820
NAT/Firewall Tranversal	no
Interface IPv4 Address *	10.0.0.1
Interface IPv4 Prefix *	30
Interface IPv6 Address *	
Interface IPv6 Prefix *	
Install Routes	yes
Traffic Selector	Subnets
Subnets *	172.16.24.0/24
Pre-shared Key *	Generate
Local Private Key	yGx2U4C+kFvLp6HRGg33LD4Xh3ljuTnU60uqHYjthno= Generate
Local Public Key *	uSa2f1uyYq6z1uOd4Y9jeSHq5PiYpUneWDcBJdtqyis=
Remote Public Key	D2OYEa0MSQR0ONI8CBaqUJJQvXo4CSqmXdE+/3x+Zxc=

** can be blank*

Apply

Figure 2: Configuration of the first router – SERVER

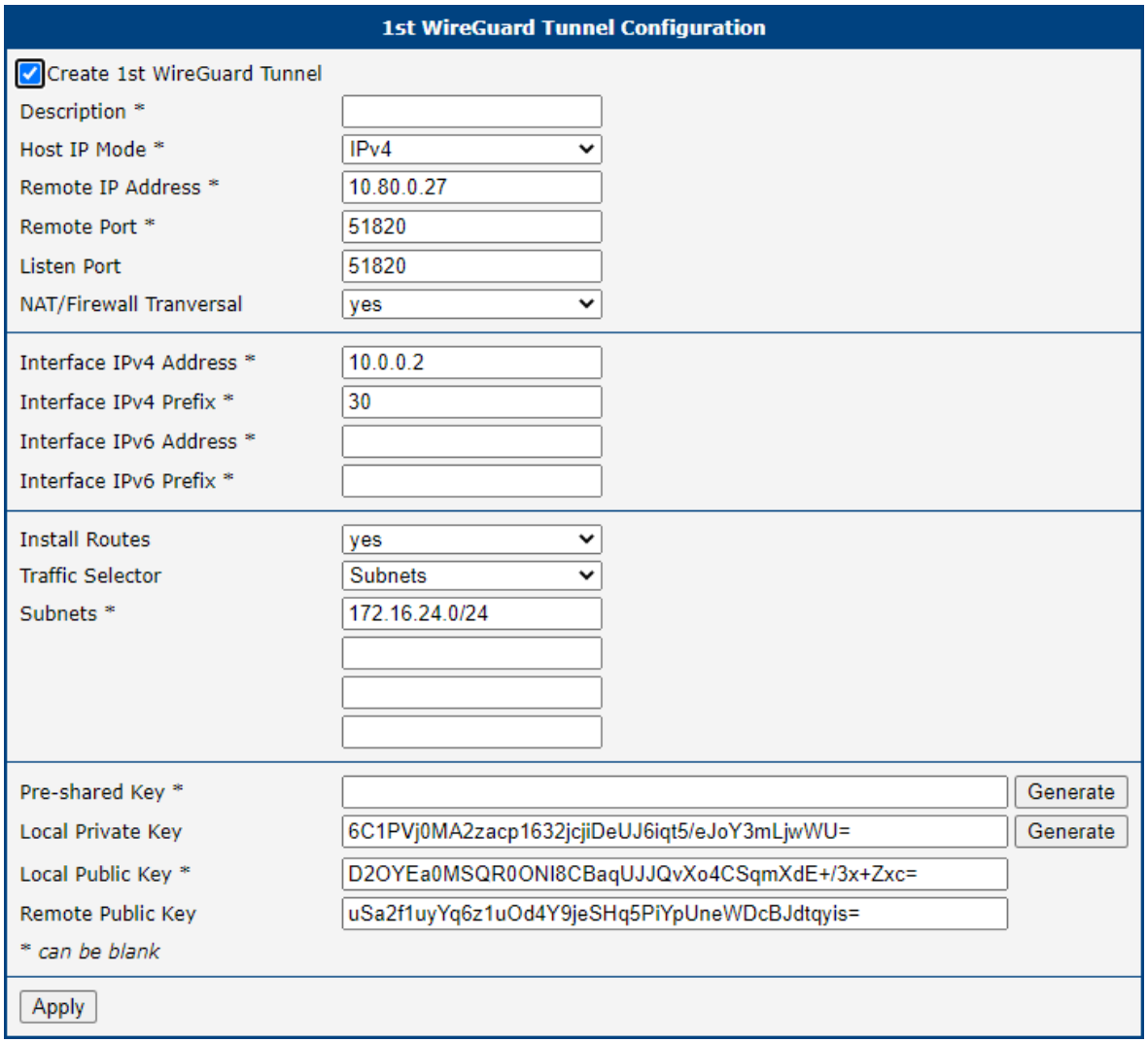

Figure 3: Configuration of the second router – CLIENT

After establishing an WireGuard tunnel, an interface wg1 and a route in the routing table of the router are displayed on the *Network Status* page.

It is also possible to check successful establishment of WireGuard tunnel in the system log (*System Log* item in menu). Listings should end with line *started*.

In the Status menu section you can find WireGuard Tunnel Status to see and confirm that the tunnel is working as should on both sides

```
wg1       Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:10.0.0.1  P-t-P:10.0.0.1  Mask:255.255.255.252
          UP POINTOPOINT RUNNING NOARP  MTU:1420  Metric:1
          RX packets:270 errors:0 dropped:0 overruns:0 frame:0
          TX packets:41 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:13396 (13.0 KB)  TX bytes:3772 (3.6 KB)
```

Route Table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
0.0.0.0	192.168.253.254	0.0.0.0	UG	0	0	0	usb0
10.0.0.0	0.0.0.0	255.255.255.252	U	0	0	0	wg1
10.64.0.0	0.0.0.0	255.255.252.0	U	0	0	0	eth0
10.65.0.0	0.0.0.0	255.255.252.0	U	0	0	0	eth1
10.66.0.0	0.0.0.0	255.255.252.0	U	0	0	0	eth2
172.16.24.0	0.0.0.0	255.255.255.0	U	0	0	0	wg1
192.168.253.254	0.0.0.0	255.255.255.255	UH	0	0	0	usb0

Figure 4: Network Status

System Log

System Messages

```
2021-09-03 00:11:35 sshd[1404]: Server listening on :: port 22.
2021-09-03 00:11:37 mwan1d[1259]: selected SIM: 1st
2021-09-03 00:11:37 mwan1d[1259]: selected APN: gprsa.agnep
2021-09-03 00:11:39 mwan1d[1259]: waiting for registration
2021-09-03 00:11:41 mwan1d[1259]: starting usbd
2021-09-03 00:11:41 usb1d[1495]: started
2021-09-03 00:11:41 usb1d[1495]: establishing connection
2021-09-03 00:11:44 usb1d[1495]: connection established
2021-09-03 00:11:44 usb1d[1495]: local IPv4 address 10.80.0.27
2021-09-03 00:11:44 usb1d[1495]: primary   DNSv4 address 10.0.0.1
2021-09-03 00:11:44 usb1d[1495]: script /etc/scripts/ip-pre-up-mwan started
2021-09-03 00:11:44 bard[988]: usb0 connection is available on mwan
2021-09-03 00:11:44 usb1d[1495]: script /etc/scripts/ip-pre-up-mwan finished
2021-09-03 00:11:44 usb1d[1495]: script /etc/scripts/ip-up-mwan started
2021-09-03 00:11:44 usb1d[1495]: script /etc/scripts/ip-up-mwan finished
2021-09-03 00:11:44 bard[988]: backup route selected: "Mobile WAN"
2021-09-03 00:11:44 bard[988]: script /etc/scripts/ip-up usb0 started
2021-09-03 00:11:46 bard[988]: script /etc/scripts/ip-up usb0 finished, status = 0x0
2021-09-03 00:11:46 dnsmasq[1328]: reading /etc/resolv.conf
2021-09-03 00:11:46 dnsmasq[1328]: using nameserver 10.0.0.1#53
2021-09-03 00:11:48 sshd[1653]: Accepted keyboard-interactive/pam for root from 10.64.0.1 port 51663 ssh2
2021-09-03 00:11:48 sshd[1653]: pam_unix(sshd:session): session opened for user root by (uid=0)
2021-09-03 08:24:40 http: user 'root' logged in from 10.64.0.1
2021-09-03 08:25:42 http: user 'root' updated configuration of WireGuard
2021-09-03 08:25:42 WireGuard: Stopping VPN tunnel wg1
2021-09-03 08:25:43 WireGuard: Starting VPN tunnel wg1
2021-09-03 08:25:43 WireGuard: Creating interface wg1
2021-09-03 08:25:43 WireGuard: Installing route 172.16.24.0/24 for wg1
2021-09-03 08:25:43 WireGuard1[22827]: started
2021-09-03 08:41:55 http: user 'root' logged in from 10.64.0.1
```

Save Log | Save Report

Figure 5: System log

WireGuard Tunnel Status

WireGuard Tunnel 1 Information

```
interface: wg1
  public key: uSa2f1uyYq6z1uOd4Y9jeSHq5PiYpUneWDcBJdtqyis=
  private key: (hidden)
  listening port: 51820

peer: D2OYEa0MSQR0ONI8CBaqUJJQvXo4CSqmXdE+/3x+Zxc=
  endpoint: 10.80.0.48:51820
  allowed ips: 10.0.0.0/30, 172.16.24.0/24
  latest handshake: 49 seconds ago
  transfer: 6.00 KiB received, 1.71 KiB sent
```

WireGuard Tunnel 2 Information

WireGuard is disabled.

WireGuard Tunnel 3 Information

WireGuard is disabled.

WireGuard Tunnel 4 Information

WireGuard is disabled.

Figure 6: WireGuard Tunnel Status for first router

WireGuard Tunnel Status

WireGuard Tunnel 1 Information

```
interface: wg1
  public key: D2OYEa0MSQR0ONI8CBaqUJJQvXo4CSqmXdE+/3x+Zxc=
  private key: (hidden)
  listening port: 51820

peer: uSa2f1uyYq6z1uOd4Y9jeSHq5PiYpUneWDcBJdtqyis=
  endpoint: 10.80.0.27:51820
  allowed ips: 10.0.0.0/30, 172.16.24.0/24
  latest handshake: 2 minutes, 7 seconds ago
  transfer: 2.07 KiB received, 7.61 KiB sent
  persistent keepalive: every 25 seconds
```

WireGuard Tunnel 2 Information

WireGuard is disabled.

WireGuard Tunnel 3 Information

WireGuard is disabled.

WireGuard Tunnel 4 Information

WireGuard is disabled.

Figure 7: WireGuard Tunnel Status for second router

3.2 WireGuard tunnel with FRR/BGP

This example shows how to run WireGuard tunnel with FRR/BGP.

FRR is Router App from Advantech and could be found and downloaded on *Engineering Portal* at icr.advantech.cz address.

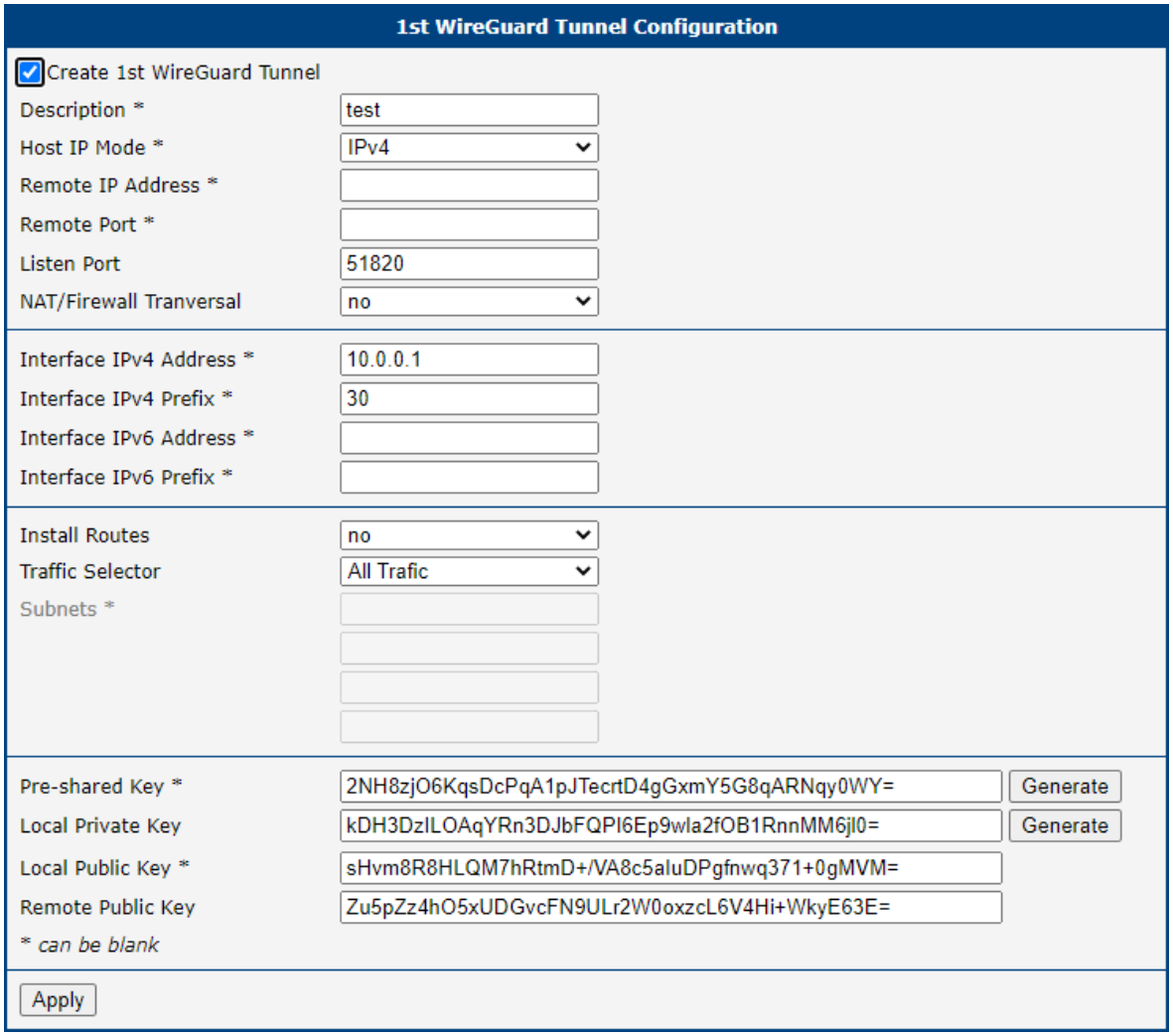

Figure 8: WireGuard Tunnel Configuration with use of FRR/BGP

ZEBRA Configuration

☑ Enable ZEBRA

```
!
! Default configuration with enabled vty
! Change password!!!
!
password conel
enable password conel
!
line vty
!
interface eth0
interface wg1
!
debug zebra events
debug zebra kernel
```

BGP Configuration

☑ Enable BGP

```
!
! Default configuration with enabled vty
! Change password!!!
!
password conel
enable password conel
!
line vty
!
log syslog
!
router bgp 11111
bgp router-id 10.0.0.1
bgp log-neighbor-changes
no bgp ebgp-requires-policy
address-family ipv4 unicast
  network 192.168.133.0/24
exit-address-family
timers bgp 3 15
!
neighbor 10.0.0.2 remote-as 22222
neighbor 10.0.0.2 disable-connected-check
neighbor 10.0.0.2 next-hop
!
debug bgp neighbor-events
debug bgp zebra
debug bgp nht
debug bgp updates
!
```

Figure 9: Zebra and BGP Configuration

Status Overview

Services

```
-----------------------------------------------------------------------
Protocol zebra is running
-----------------------------------------------------------------------
FRRouting 7.5 (Router).
Router# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup

K>* 0.0.0.0/0 [0/0] via 192.168.253.254, usb0, 02:55:35
C>* 10.0.0.0/30 is directly connected, wg1, 02:55:34
C>* 89.24.1.79/32 is directly connected, usb0, 02:55:36
B>* 192.168.1.0/24 [20/0] via 10.0.0.2, wg1, weight 1, 02:54:42
C>* 192.168.7.0/24 is directly connected, eth1, 02:55:52
K>* 192.168.253.254/32 [0/0] is directly connected, usb0, 02:55:35
Router# show ipv6 route
Codes: K - kernel route, C - connected, S - static, R - RIPng,
       O - OSPFv3, I - IS-IS, B - BGP, N - NHRP, T - Table,
       v - VNC, V - VNC-Direct, A - Babel, D - SHARP, F - PBR,
       f - OpenFabric,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup

C>* 64:ff9b::/96 is directly connected, nat64, 02:55:35
C>* fd00::/64 is directly connected, eth1, 02:55:52
C * fe80::/64 is directly connected, nat64, 02:55:35
C>* fe80::/64 is directly connected, eth1, 02:55:52


-----------------------------------------------------------------------
Protocol nhrp is stopped
-----------------------------------------------------------------------
-----------------------------------------------------------------------
Protocol bgp is running
-----------------------------------------------------------------------
Router# show ip bgp
BGP table version is 1, local router ID is 10.0.0.1, vrf id 0
Default local pref 100, local AS 11111
Status codes:  s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Nexthop codes: @NNN nexthop's vrf id, < announce-nh-self
Origin codes:  i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.1.0/24   10.0.0.2                 0             0 22222 i
   192.168.133.0/24 0.0.0.0                  0         32768 i

Displayed  2 routes and 2 total paths


-----------------------------------------------------------------------
Protocol isis is stopped
-----------------------------------------------------------------------
```

Figure 10: FRR/BGP Status Overview

WireGuard Tunnel Status

WireGuard Tunnel 1 Information

```
interface: wg1
  public key: sHvm8R8HLQM7hRtmD+/VA8c5aIuDPgfnwq371+0gMVM=
  private key: (hidden)
  listening port: 51820

peer: Zu5pZz4hO5xUDGvcFN9ULr2W0oxzcL6V4Hi+WkyE63E=
  preshared key: (hidden)
  endpoint: 176.102.145.59:51820
  allowed ips: 0.0.0.0/0, ::/0
  latest handshake: 26 seconds ago
  transfer: 743.09 KiB received, 624.16 KiB sent
```

WireGuard Tunnel 2 Information

WireGuard is disabled.

WireGuard Tunnel 3 Information

WireGuard is disabled.

WireGuard Tunnel 4 Information

WireGuard is disabled.

Figure 11: WireGuard Tunnel Status

3.3 WireGuard with FRR/staticd

Some customers may want more than 4 static routes. In this case is possible to use staticd.

FRR is Router App from Advantech and could be found and downloaded on *Engineering Portal* at icr.advantech.cz address.

STATIC Configuration

☑ Enable STATIC

```
!
! Default configuration with enabled vty
! Change password!!!
!
password conel
enable password conel
!
line vty
!
ip route 172.16.0.0/16 wg1
ip route 172.20.0.0/16 wg1
ip route 172.24.0.0/16 wg1
ip route 192.168.10.0/24 wg1
ip route 192.168.20.0/24 wg1
ip route 192.168.30.0/24 wg1
```

Apply

Figure 12: Static Configuration

ZEBRA Configuration

☑ Enable ZEBRA

```
!
! Default configuration with enabled vty
! Change password!!!
!
password conel
enable password conel
!
line vty
!
interface eth0
interface wg1
!
debug zebra events
debug zebra kernel
```

Apply

Figure 13: Zebra Configuration

Status Overview

Services

```
-----------------------------------------------------------------------
Protocol zebra is running
-----------------------------------------------------------------------
FRRouting 7.5 (Router).
Router# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup

K>* 0.0.0.0/0 [0/0] via 192.168.253.254, usb0, 03:59:49
C>* 10.0.0.0/30 is directly connected, wg1, 03:59:48
C>* 89.24.1.79/32 is directly connected, usb0, 03:59:50
S>* 172.16.0.0/16 [1/0] is directly connected, wg1, weight 1, 01:02:19
S>* 172.20.0.0/16 [1/0] is directly connected, wg1, weight 1, 01:02:19
S>* 172.24.0.0/16 [1/0] is directly connected, wg1, weight 1, 01:02:19
C>* 192.168.7.0/24 is directly connected, eth1, 04:00:06
S>* 192.168.10.0/24 [1/0] is directly connected, wg1, weight 1, 01:02:19
S>* 192.168.20.0/24 [1/0] is directly connected, wg1, weight 1, 01:02:19
S>* 192.168.30.0/24 [1/0] is directly connected, wg1, weight 1, 01:02:19
K>* 192.168.253.254/32 [0/0] is directly connected, usb0, 03:59:49
Router# show ipv6 route
Codes: K - kernel route, C - connected, S - static, R - RIPng,
       O - OSPFv3, I - IS-IS, B - BGP, N - NHRP, T - Table,
       v - VNC, V - VNC-Direct, A - Babel, D - SHARP, F - PBR,
       f - OpenFabric,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup

C>* 64:ff9b::/96 is directly connected, nat64, 03:59:49
C>* fd00::/64 is directly connected, eth1, 04:00:06
C * fe80::/64 is directly connected, nat64, 03:59:49
C>* fe80::/64 is directly connected, eth1, 04:00:06
```

Figure 14: WireGuard Status Overview

4. Related Documents

[1] Advantech Czech: **SmartFlex Configuration Manual** (MAN-0023-EN)
[2] Advantech Czech: **SmartMotion Configuration Manual** (MAN-0024-EN)
[3] Advantech Czech: **SmartStart Configuration Manual** (MAN-0022-EN)
[4] Advantech Czech: **ICR-3200 Configuration Manual** (MAN-0042-EN)
[5] Advantech Czech: **FRR Router App Aplication Note** (APP-0100-EN)

Product-related documents can be obtained on *Engineering Portal* at icr.advantech.cz address.