Using Katzenpost with NAT and Tor

Table of Contents

Katzenpost servers can be used from behind network address translation (NAT) devices. This applies to mix nodes, gateway nodes, service nodes, and directory authority nodes (dirauths). Port forwarding and other network configuration details depend on how you are hosting your servers and the type of router you use.

Some hosting scenarios, such as the use of an Amazon Elastic Compute Cloud (Amazon EC2) instance, require no manual port forwarding. A Katzenpost node running on an EC2 instance with default network settings listens on its internal IP address yet can receive connections from publicly routed IP addresses. For home and small business routers, typical default policy is to block inbound connections from public addresses. In this scenario, you need to configure port-forwarding to the appropriate internal IP address and port of the destination node.

Router and LAN host configuration in a NAT topology are beyond the scope of this topic. NAT was defined in 1994 in RFC 1631 [https://www.rfc-editor.org/rfc/rfc1631] and tutorials are widely available.

Note

Katzenpost does not support NAT penetration protocols such as NATPMP [https://www.rfc-editor.org/rfc/rfc6886], STUN [https://www.rfc-editor.org/rfc/rfc5389], and TURN [https://www.rfc-editor.org/rfc/rfc5766].

Addresses and *BindAddresses*

Any Katzenpost server node can be configured to support NAT and similar network topologies (such as Tor) that traverse public and private network boundaries. In a direct network connection, the address defined in the server *Addresses* parameter defines the addresses on with the node listens for incoming connections, and which are advertised to other mixnet components in the PKI document. By supplying the optional *BindAddresses* parameter, you can define a second address group: internal addresses that are *not* advertised in the PKI document. This is useful for NAT and Tor scenarios.

The following table shows the details for these two parameters. For more information about mixnet node configuration, see Components and configuration of the Katzenpost mixnet [https://katzenpost.network/docs/admin_guide/components.html].

Table 1. *Addresses* and *BindAddresses* parameters

Parameter	Required	Description
Addresses	Yes	Specifies a list of one or more address URLs in a format that contains the transport protocol, IP address, and port number that the server will bind to for incoming connections. This value is advertised in the PKI document. Katzenpost supports URLs with that start with either `tcp://` or `quic://` such as: `["tcp://192.168.1.1:30001"]` and

Parameter	Required	Description
		`["quic://192.168.1.1:40001"]`. Onion addresses (transport protocol `onion://`) are also supported on some node types if *BindAddresses* is also present. Overridden by *BindAddresses*; see below.
BindAddresses	No	If **true** (that is, if the parameter is present), this parameter allows you to set internal listener addresses that the server will bind to and accept connections on, but that are not advertised in the PKI document. This parameter is used to configure NAT and Tor support.

For example, a mix node configuration file with a direct Internet connection, or with transparent NAT hosting such as that provided by Amazon EC2, contains a `Server` section specifying only the *Addresses* parameter, as in the following listing:

```
[Server]
  Identifier = "mix1"
  WireKEM = "xwing"
  PKISignatureScheme = "Ed448-Dilithium3"
  Addresses = ["tcp://127.0.0.1:30010",]
  MetricsAddress = "127.0.0.1:30012"
  DataDir = "/voting_mixnet/mix1"
  IsGatewayNode = false
  IsServiceNode = false
```

Importantly, the value of *Addresses*, which the node advertises to other peers, is its loopback address, 127.0.0.1, and a specified port. This means that there is no publicly routable address for this node, and that traffic arriving at the node is delivered there by external infrastructure, presumably a configured LAN. In all of these examples, nodes have necessarily been assigned a LAN addresses, but these addresses play no role in Katzenpost server configurations. Rather, for portability, the configurations use the logically equivalent loopback address.

In contrast to the direct connection, the next example shows the configuration file for a similar node behind NAT, containing a `Server` section that specifies both the *Addresses* and *BindAddresses*parameters:

```
[Server]
  Identifier = "mix1"
  WireKEM = "xwing"
  PKISignatureScheme = "Ed448-Dilithium3"
  Addresses = ["tcp://203.0.113.10:30010"]
  BindAddresses = ["tcp://127.0.0.1:30010"]
  MetricsAddress = "127.0.0.1:30012"
  DataDir = "/voting_mixnet/mix1"
  IsGatewayNode = false
  IsServiceNode = false
```

Notice that the internal listening address remains unchanged, but it's has been shifted to the new *BindAddresses* parameter. In addition, the value of the *Addresses* parameter has been changed to a public address (the NAT router's WAN address), including a port. It is assumed that the router is configured to forward traffic routed to this address:port to the mixnet node over its LAN.

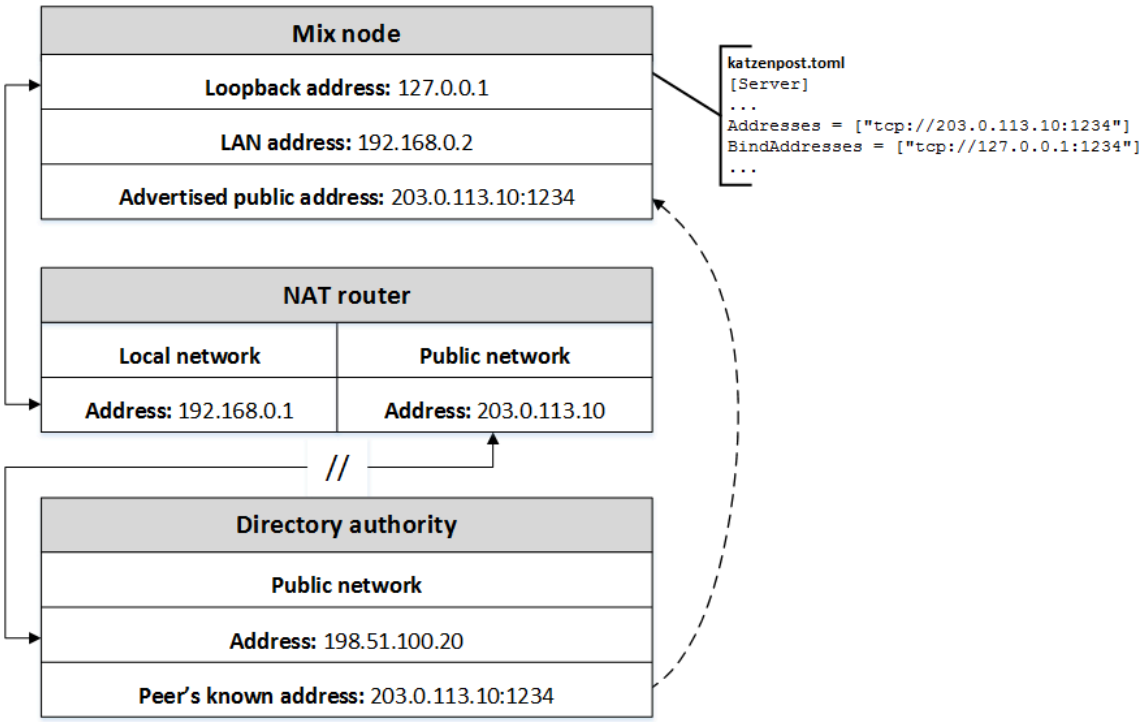
Application scenarios

The following sections illustrate supported Katzenpost topologies that make use of the *BindAddresses* parameter.

Peer-to-peer connections involving NAT

In this scenario, a mix node behind NAT listens on local addresses for connections, while advertising a public address and port to its peer, a directory authority, that is assumed to have a publicly routable address.

Figure 1. Configuring a mix node behind NAT to be available to a dirauth



Enlarge diagram [https://katzenpost.network/docs/admin_guide/pix/peer-to-peer-nat.png]

Key observations

- The configuration file on the NATed mix node is `katzenpost.toml`.
- The relevant section of the configuration file is `[Server]`.
- The *`Addresses`* parameter specifices the publicly routable address (203.0.113.10) and a port (1234) over which the mix node can be reached from the Internet. This value is periodically advertised in the PKI document to other components of the mix network.
- The *`BindAddresses`* parameter specifies the loopback address (127.0.0.1) and a port (1234) on which the node listens for incoming Sphinx packets from peers.
- The mix node also has a LAN address, 192.168.0.2, which does not appear in the Katzenpost configuration, but is logically equivalent to the node's loopback address, 127.0.0.1.
- The NAT router has two configured addresses, public address 203.0.113.10, and LAN address 192.168.0.1.
- The NAT device routes traffic for 203.0.113.10:1234 to the LAN address and port of the mix node, 192.168.0.2:1234, and therefore to the configured listener bound to 127.0.0.1:1234.

Client-to-gateway connections involving NAT

In this scenario, a gateway node behind NAT listens on local addresses for connections from the Internet, while advertising a public address and port to a client application that is assumed to have a publicly routable address.

Figure 2. Configuring a gateway behind NAT to be available to a client

Enlarge diagram [pix/client-gateway-nat.png]

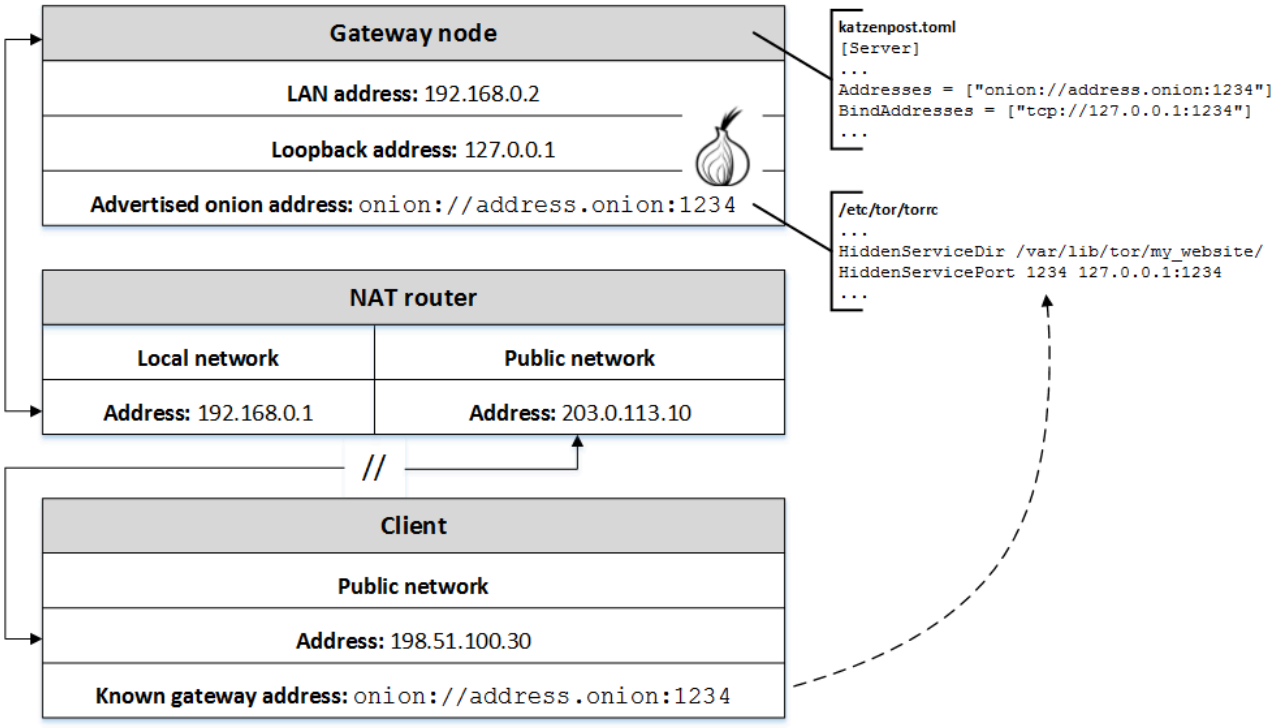
Key observations

- The configuration file on the NATed gateway node is `katzenpost.toml`.
- The relevant section of the configuration file is `[Server]`.
- The *`Addresses`* parameter specifices the publicly routable address (203.0.113.10) and a port (1234) over which the gateway node can be reached from the Internet. This value is periodically advertised in the PKI document to other components of the mix network.
- The *`BindAddresses`* parameter specifies the loopback address (127.0.0.1) and a port (1234) on which the node listens for incoming Sphinx packets from clients.
- The gateway node also has a LAN address, 192.168.0.2, which does not appear in the Katzenpost configuration, but is logically equivalent to the node's loopback address, 127.0.0.1.
- The NAT router has two configured addresses, public address 203.0.113.10, and LAN address 192.168.0.1.
- The NAT device routes traffic for 203.0.113.10:1234 to the LAN address and port of the gateway node, 192.168.0.2:1234, and therefore to the configured listener bound to 127.0.0.1:1234.

Client-to-gateway connections over Tor

In this scenario, a gateway node behind NAT listens on local addresses for connections from the Internet, while advertising an onion address and port to a client application that is assumed to have a publicly routable address. Apart from the use of an onion address, this scenario is identical to the previous one.

Figure 3. Conguring a gateway to be availabe to a client over Tor



Enlarge diagram [https://katzenpost.network/docs/admin_guide/pix/client-gateway-onion.png]

Key observations

- The configuration file on the NATed gateway node is `katzenpost.toml`.
- The relevant section of the configuration file is `[Server]`.
- The gateway node exposes itself to the network as an onion (hidden)service bound to the loopback address (127.0.0.1) and a port (1234). This requires a Tor daemon to be running on the node and for the code snippet from the Tor configuration file `/etc/tor/torrc` to contain the lines shown.
- The *`Addresses`* parameter specifices the onion address (`address.onion`) and a port (1234) over which the gateway node can be reached by way of the Tor network. This value is periodically advertised in the PKI document to other components of the mix network.
- The *`BindAddresses`* parameter specifies the loopback address (127.0.0.1) and a port (1234) on which the node listens for incoming Sphinx packets from clients.
- The gateway node also has a LAN address, 192.168.0.2, which does not appear in the Katzenpost configuration, but is logically equivalent to the node's loopback address, 127.0.0.1.
- The NAT router has two configured addresses, public address 203.0.113.10, and LAN address 192.168.0.1, which in this case play no role in the Katzenpost configuration.
- The NAT device routes traffic for 203.0.113.10:1234 to the LAN address and port of the gateway node, 192.168.0.2:1234, and therefore to the configured onion service listener bound to 127.0.0.1:1234.