ZeroTier Version 2 Protocol

Monica Moniot

March 14, 2023

1 Definitions

For clarity's sake, when I use the name ZeroTier, unless otherwise specified, I am referring to Version 2 (V2) of the ZeroTier protocol. This is to disambiguate between the V1 ZeroTier protocol, and ZeroTier Inc., the organization which maintains both. The goal of ZeroTier in essence is to provide a secure, reliable and scalable protocol for implementing a computer network with no tiers, also known as a flat network.

Definition 1 (Node)**.** A *node* in ZeroTier is a client machine running the ZeroTier protocol. A node is capable of making peer-to-peer, end-to-end encrypted ethernet connections with any other node, so long as it knows the address of the other node, and the other node chooses to respond to its connection. Public keys are used to identify and authenticate nodes.

The ZeroTier protocol emulates the ethernet protocol for the sake of being as low down on the protocol stack as practical, improving compatibility and limiting complexity.

Definition 2 (Address)**.** The *address* of a node in ZeroTier is a uniquely identifying 40-bit number derived from the hash of the node's public key.

Definition 3 (Identity)**.** The *identity* of a node in ZeroTier consists of that node's public key and the matching address.

Definition 4 (Path)**.** A *path* is the physical link that is used for one node to send network packets to another. Generally this is a direct link over UDP, in which case the path is fully specified by the IP address and open port of the destination node. Paths are symmetric, meaning they allow for two way communication between source and destination. Paths are automatically found and established using DNS-like name resolution and NAT hole punching.

2 ZeroTier Secure Session Protocol

ZeroTier uses a FIPS compliant implementation of Noise_XKpsk3 as its transport protocol. To understand the following section, we will assume a basic familiarity with the Noise protocol, documented here https://noiseprotocol.org/noise.html. We will primarily be explaining the ZeroTier transport protocol in relation to the specification of the noise protocol. Noise has a very particular naming scheme for instanciations of it, and our full protocol name is

`Noise_XKpsk3_P384_AESGCM_SHA384_hybridKyber1024`.

We only implement the above protocol with those static choices of cryptographic primitives. `Noise_XKpsk3` specifies we are implementing Noise XK with a preshared symmetric secret used on the third round of our handshake. `P384` specifies we are using eleptic curve P384 as our FIPS compliant Diffie-Helman (DH) primitive. `AESGCM` specifies we are using the AES block cipher in GCM mode as our symmetric primitive. `SHA384` is our FIPS compliant hash function. And finally, `hybridKyber1024` specifies we are adding hybrid encryption onto Noise using an ephemeral Kyber1024 DH handshake.

There is one and only one difference between the ZeroTier Secure Session Protocol and Noise_XKpsk3, and that is our choice of key derivation function (KDF). Noise specifies the use of HKDF, https://eprint.iacr.org/2010/264.pdf, but despite it's strength, it has not yet been approved under FIPS. This gave us two options, we could either modify HKDF to pass any output key through an addition FIPS KDF, or we could entirely replace HKDF with a FIPS KDF. We decided on the later, using NIST's currently recommended KDF in HMAC counter mode, https://csrc.nist.gov/publications/detail/sp/800-108/archive/2008-11-06. To be specific, our implementation of 'MixKey(*input_key_material*)' is as follows:

- Sets $ck = \text{HMAC}(ck,\ input\_key\_material)$.
- Sets $temp\_k = \text{HMAC}(ck,\ [1]_2 \mid\mid key\_label \mid\mid 0x00 \mid\mid 0 \mid\mid [256]_2)$.
- Truncates $temp\_k$ to 256 bits (64 bytes).
- Calls InitializeKey($temp\_k$)

where key_label is a byte whose value is unique for each $temp\_k$ that is produced for encryption during the Noise handshake.

And our implementation of 'Split()' is as follows:

- Sets $temp\_k1 = \text{HMAC}(ck,\ [1]_2 \mid\mid b'A' \mid\mid 0x00 \mid\mid 0 \mid\mid [256]_2)$.
- Sets $temp\_k2 = \text{HMAC}(ck,\ [1]_2 \mid\mid b'B' \mid\mid 0x00 \mid\mid 0 \mid\mid [256]_2)$.
- Truncates $temp\_k1$ and $temp\_k2$ to 256 bits (64 bytes).
- Creates two new CipherState objects c1 and c2.
- Calls c1.InitializeKey(temp_k1) and c2.InitializeKey(temp_k2).
- Returns (c1, c2).

3 Beginning of unfinished, outdated documentation (below)

4 ZeroTier Rendezvous Protocol

All nodes start only knowing a few paths to the root servers. ZeroTier is able to bootstrap this into a peer-to-peer connection with any other node. Given the address of a destination node, a source node will contact a random root server to request the identity of the node with that address. The root server will look up the destination node's identity and return it. Once the source node has its identity, it will then choose another random root server to request a rendezvous with the destination node. The root server will then search through all known paths to both the destination and the source nodes, and choose a pair most likely to succeed in back and forth communication. To the source node it will send the destination's path, and to the destination node it will send the source's path. Upon receipt each node will attempt to contact the other through the given path. If either contact is successful the recipient will save the successful path. From then on both nodes will choose the highest quality known path to communicate with each other. Upon receipt of any packet from a node, the path it was received over is either added to the list of known paths, or if it already exists it is updated help with quality determination.

NAT Hole Punching

The reason two paths are simultaneously attempted to establish contact is to open up NAT gateways and stateful firewalls. This is NAT hole punching, and it is essential for enabling two peers to communicate without having to establish a shared, open IP port first. The pair of paths is specifically chosen to increase the odds of a successful hole punch through most NAT gateways and firewalls. If contact was unsuccessful, traffic between the nodes will be routed through the root servers instead, but rendezvous will be periodically reattempted.

5 ZeroTier Client Protocol

When ZeroTier is installed into a host machine, it will generate a random public and private key. The private key is saved to the host machine. The public key is used to generate an address and identity. This identity is submitted to a random root server, and if the root server confirms that the node's address is not in use, it is saved to both the host machine and the root server. If it is in use the host machine will generate a new keypair and try again.

Next the host machine will ask for a network id from the user. When this is given the host machine will rendezvous with the network controller for that network. Upon a successful rendezvous the node will request to join the network controller's network. If the request is approved, the network controller assigns it a virtual IP address, and sends it any network-specific certificates.

ZeroTier offers ARP and NDP emulation to help client machines map a virtual IP address on a given ZeroTier network to the ZeroTier address of its node. However these protocols are not emulated 1 to 1, in particular because ARP scales poorly on large networks. Instead, when a host machine wishes to connect to a new virtual IP address, it multicasts its request to connect directly to the owner of that virtual IP address, the destination node. See Section 6 for more details on ZeroTier multicast. The destination node will post the received ARP or NDP ethernet frame to the host OS. The host OS will decide whether or not to reply according to ARP or NDP. It is after the source receives a reply that it will request rendezvous with the destination and establish a session. Identities and paths are always cached so in the future the host machine is able to resolve the IP address to a path to the destination node without the help of the network controller or a root server.

6 ZeroTier Multicast Protocol

Definition 5 (Multicast Group)**.** A *multicast group* is a set of nodes identified with a unique MAC address and optionally a 32-bit number, called "additional distinguishing information", or ADI.

Nodes may send "multicast-like" packets to any other node on the same network as them. This packet contains the MAC address and optional ADI of a multicast group they would like to join. Every node locally records what multicast-likes they have received, and considers a node to be a member of a multicast group if they have sent it a multicast-like packet. Nodes automatically update known peers about their group membership status through multicast-like packets.

Nodes may send "multicast-gather" packets to any other node on the same network. This packet contains the MAC address and ADI of a multicast group. It also contains a number called the "gather limit". A node will reply to a multicast-gather packet with a list of the addresses of every node it considers a member of that multicast group. This list will be of length at most the gather limit. No reply is sent if the list is empty.

Nodes may send "multicast-frame" packets to any other node on the same network. This packet again contains the MAC address and ADI of a multicast group, a gather limit, and an ethernet frame. A node will reply to this packet the same as multicast-gather. In addition, if the receiving node is a member of the multicast group specified by the packet, the node will consider the ethernet frame as addressed to it and receive it.

The above protocol allows for all nodes on a given network to very quickly be updated on the members of any multicast group that the node wants to join or send frames to. This allows multicasts to be performed peer-to-peer. However, in some networks, peer-to-peer multicast is not as efficient as hub-and-spoke multicast. In those cases one may choose to use the replicator protocol for their network, described below.

Definition 6 (Multicast Replicator)**.** A node may become a *multicast replicator* if it has been given permission to by a network controller. Multicast replicators will replicate any multicast-frame packets they receive and forward them to every member of the multicast group specified by the packet. Nodes on a network will send their multicast frames to a multicast replicator if one exists, instead of doing sender-side replication. If multiple exist the node selects the one with the lowest path latency.

MAC address `0xffffffffffff` is reserved for the "broadcast" multicast group. When any node joins a network, they automatically like this group with an ADI value of their assigned IPv4 address. This allows a multicast frame to be sent to all members of a network, or to only a member with a specific IPv4 address.

ARP emulation is performed by multicasting to group `0xffffffffffff` with ADI the IPv4 address being looked up. IPv6 addresses in ZeroTier always contain 24-bits that identify a multicast group. Thus NDP emulation is more straightforward, as a node can multicast directly to that group to lookup that IPv6 address. If a node is assigned an IPv6 address it will automatically like the group identified by it.

Security Considerations

The only access control applied to a multicast group is whether or not a node has permission to join a network. In addition, when a frame is replicated for multicast, it is the plaintext of the frame that is replicated, and then encrypted under only the peer-to-peer key of each destination node. As such the plaintext of all multicast traffic over a network is effectively fully visible to all members of the network.

7 ZeroTier Heartbeat Protocol

Every node creates a local database of ever peer it has rendezvous with. This database stores the peer's identity and a table of up to 64 known paths to the peer. Several quality of service metrics and timestamps are included in this database to help with making decisions throughout the ZeroTier protocol about which peers to contact or which paths to use.

Every 14 seconds, for every known path, an “echo” packet is sent over it to the peer. Nodes are expected to reply to echo packets over the same path the packet was received over. It is when the reply to the echo is received that the node can confirm that the path is still alive. If any packet is received over a given path, the expiry timestamp on that path is updated to the current time. If the current time and the expiry timestamp on a path ever differ by over 243 seconds, the path is considered expired and is deleted from the table of known paths.

The rendezvous protocol is specifically designed to facilitate hole punching through most NAT gateways. However some NAT gateways keep these holes open for very short periods of time unless the connection through the hole is actively used. The heartbeat protocol is designed to combat this and keep paths open and alive for as long as possible. This is especially important for keeping paths to the root servers alive, otherwise it would not be possible for nodes to initiate new connections.

8 ZeroTier Network Certificates

Definition 7 (Network Certificate)**.** *Network certificates* are sequences of bytes issued by a network controller that specify certain permissions or capabilities a node has on the network controller's network. They are cryptographically signed by the network controller with its private key, allowing all network members to verify authenticity. All network certificates include a creation timestamp and lifetime written by the controller. Certificates also include a network id and an address of a node. These specify respectively, which network a certificate was issued on and what node owns this certificate. Nodes will locally verify all of these fields on any received certificate to confirm its validity.

Nodes will never use their local clock to determine whether a certificate has expired, instead they are compared relative to each other based on the type of certificate received. Thus it is important that the network controller has a secure and accurate source of unix time. All other network nodes can securely check for expiry without a secure source of time.

Certificates, once created, are usually immediately sent to their owners, and no one else. This places the bandwidth cost of distributing certificates onto only the owner, and the owner only needs to send these certificates to nodes it intends to communicate with.

Definition 8 (Certificate of Membership)**.** A *Certificate of Membership* is the simplest form of certificate. It contains, in addition to the information described above, a hash of the full identity of a node. It determines whether or not other nodes consider the owner to be a member of the network specified by the certificate. A node is authorized on a network if it has a valid certificate of membership.

It is the responsibility of each network member to present its certificate of membership to any other member immediately after completing the transport protocol handshake. Members will ignore most forms of communication, especially ethernet frames, over a new connection until a valid certificate of membership is presented to it.

Definition 9 (Certificate of Ownership)**.** A *Certificate of Ownership* contains an IPv4 address and an IPv6 address. A node is considered as owning an IPv4 or IPv6 address if it has a Certificate of Ownership containing those addresses.

It is required that each network member present their certificate of ownership at the same time as their certificate of membership, to proof that they have not spoofed their IPv4 or IPv6 address. Network members will again ignore connections that do not present valid certificates matching the IP address they were expecting to connect to.

Definition 10 (Capability)**.** A *Capability* is a certificate that contains byte-code representing the network rules for the specified network. A node that has received this certificate will run the code within every time it attempts to send or receive a packet containing an ethernet frame. This code is the foundation of the ZeroTier rules engine. To summarize briefly, it will decide what action should be taken with the new packet, including potentially dropping it entirely.

Definition 11 (Tags)**.** *Tags* are key-value pairs that are contained in a tag certificate. These key-value pairs by themselves carry no meaning, but can be read by the rules engine to create custom, capability-based permissions on a network. A single tag certificate can contain multiple tags, and multiple tags distributed under a single certificate can only be broken into multiple certificates by the network controller.

Definition 12 (Revocation)**.** A *Revocation* is a certificate that revokes membership to a given network. It contains no extra information, but unlike other certificates, it only contains one timestamp field and no lifetime. Instead the timestamp field is the time at which the revocation expires. Any certificate of membership issued to the specified node before the expiration timestamp will be considered revoked, but any issued after are considered valid. This allows a network controller to immediately revoke all existing certificates of membership for a given node, while still potentially allowing it to reauthorize the node later.

Instead of being sent to the node being revoked, revocations are instead sent from the network controller to all members of the network. If a member sees a revocation it has not previously recorded, it will also send a copy of it to all other members it knows of. This is a rumor mill algorithm that guarantees that so long as a node is online and connected to the network through some path, it will receive the revocation.

Security Considerations

If a node was offline while a revocation was being distributed, it will not be able to receive it. In addition, only certificates of membership can be revoked with a revocation, all other certificates will last until they expire. They will continue to be valid even if the node is revoked from the network and then reauthorized later. Advanced network administrators have to be mindful of the fact that network members can pick and choose which certificates they present to which nodes. This makes certificates which limit or suspend permissions on a network very difficult to implement. The certificate system will be completely overhauled and hardened in V2 of ZeroTier.