Overview of Zero-trust Packet Routing

Danny Hillis, September 2022
(Updated May 21, 2025)

1. The Problem that Needs Solving

The Internet was designed to allow everything to communicate with everything else, and that created a lot of security problems. Internet Protocol was originally designed with the explicit assumption that security was not the responsibility of the network. Instead, all security was left to the endpoints. Everything that connected to the Internet was expected to protect itself from harmful communication. This proved to be more difficult than anticipated.

Once it was understood that it was *not* desirable for everything to be able to communicate with everything else, enterprises began to section off parts of their own networks by adding firewalls to block undesired communication. The limits on communication emerged from the firewall rules and the topology of the network. Since there is no way to know the actual source of an IP packet, the rules were often imperfect attempts to defend against specific types of attacks. As firewall rules accumulated over time, the original purposes of specific rules were often forgotten. The protections that emerged from the interaction of the accumulated firewall rules, evolving network topology and the security mechanisms of the endpoints were poorly understood and often flawed. Today, malicious actors discover and exploit these security flaws on a regular basis, leaving enterprises vulnerable to cyberattacks. This state of network security is already unacceptable, and it is getting worse.

An enterprise should be able to understand and control the communication policies of its own network. The policies should be clearly described in one place, in a way that humans can easily understand and audit them. It should be easy to prove that the intended policies will always be enforced.

Zero-Trust Packet Routing (ZPR, pronounced zip'-er) is a way to accomplish these goals. ZPR provides a way to describe communications policies and a way to reliably enforce them within the network. Since policies are enforced by the network itself, they provide an additional layer of security that is independent of any existing protection provided by endpoints and firewalls.

This document gives an overview of ZPR, illustrating how policies are defined and how they are enforced.

2. How Policies are Defined

2.1. The ZPR Policy Language

The ZPR policy language, called ***ZPL,*** is optimized for clarity to human policy writers and auditors. All communications policies in the ZPL are stated in terms of the attributes of the communicators, and the circumstances at the time of communication. The communicators involved may be endpoints, users, and/services. In ZPR these all have authenticatable ***identities***, which have associated attributes. ZPL policies are stated in terms of the attributes and the ***circumstances*** at the time of the communication, such as the time of day or how much data has recently been communicated.

All communication on a ZPR network must be specifically allowed by a permission statement. To make the policies easier to understand and audit, all permission statements in ZPL are positive allowances, and there is no way for one statement to override another. Any limitation to a permission must be stated in the permission statement itself. The advantage of this is that the policy consequences of each statement can be understood independently of context.

For example, a permission statement might allow the network to communicate classified information between government employees, with the proper clearances, connecting through authorized devices, while they are located within a government facility, unless they have exchanged more than 100 gigabytes of information in the last 24 hours.

In this example, the communication flows are associated with both a user and an endpoint device. The employment status and clearances are attributes of the users, and the approval status and location are attributes of the device. The amount of data that has been recently communicated is a circumstance at the time of communication.

Policies written in ZPL not only describe what is permitted; they also describe the intent of the combined permissions, including what is intended to be disallowed. This is accomplished with ***assertions*** of intent. For example, an assertion might state that no classified information should be allowed to be communicated to endpoints located outside of a classified facility. Such an assertion does not change the permissions, it merely specifies a way to automatically check that the permissions are consistent with the intent. If they are not, this prevents a set of policies from being adopted. If unintended communication is attempted, assertion can create an alert and block the communication.

ZPL can express any network policy that could be implemented by conventional methods such as firewalls, NATs, VLANs, etc. but it can also express much more. For example, it can express policies that depend on global measures of communication, such as limits on the amount of information an endpoint, service or user can move out of a facility during a given time period.

ZPR is designed so that every policy expressed in ZPL can be enforced by the ZPR network itself. How this enforcement works is described in section 3.

2.2. Attributes and Local Circumstances

Attributes describe properties associated with a user, device, or service. Typical attributes include roles, group membership, capabilities, identification numbers and public authentication keys. ZPL allows different classes of endpoint, users or services to be defined with specific attributes or attribute values. An attribute always has a name and sometimes it has a value, or a set of values associated with that name. An attribute without a value is called a ***tag***. Such values may be mutable or immutable. For example, a user that is an employee might have an attribute named **Employee-ID-number** that has a value that is a permanently assigned valid ID number. An employee might also have an attribute named **mobile-devices** with a mutable value that is set of mobile devices currently issued to that employee, such as {**cellphone-6789**, **laptop-2468**, **tablet-1357**}. In this case, devices that are in this set may have attributes of their own.

The ***identity*** of a user, device or service is a unique string that is immutable. All identities are authenticatable, for example through a cryptographic certificate or login authentication service. Permissions are normally not stated in terms of identities, but rather the attributes of

those identities. The identities themselves are used for determining associated attributes and for logging.

Circumstances are similar to attributes in that they have a name and a value, but they are associated with the circumstances at the time of communication rather than with an identity. The time, date, and measures of the amount of data previously communicated are examples of such circumstances.

The values of attributes are determined from ***trusted sources***, such as a trusted LDAP or Active Directory service. Circumstances are evaluated at the time they are used to make a decision.

ZPR concept of identity is described in more detail in ZPR-RFC-16.

2.3. ZPL Policy Examples

In the ZPR policy language (ZPL), statements express permissions, assertions of intent, or definitions of new classes.

A permission for users with an attribute tag top-secret could be expressed in the policy language like this:

```
Allow top-secret users to access top-secret services.
```

An assertion that no internet-gateway should be allowed to access internal services could be expressed this:

```
Never allow internet-gateways to access internal services.
```

In these examples top-secret and internal are tags, and internet-gateway is a class of devices that is defined by its attributes.

ZPL also supports assertions about the allowable values of attributes. ZPL is described in more detail in ZPR-RFC-15.

2.4. Policy for an Organization

Within a large organization different parts of network policy may be managed by different people. Since ZPL statements are not modified by context, ZPL policy descriptions can be hierarchically combined from each source to create a global policy for a network. All of the permissions are combined and tested against the constraints of enterprise-wide assertions. Since permissions are additive, they can never conflict with other permissions but if they conflict with assertions, that conflict must be resolved before the new policy is accepted. Source control tools can be used for combining ZPL-format source files maintained by different parts of the organization.

The combined policy file includes all the policy permissions and assertions in a form that is convenient to read and audit. During compilation of the policy descriptor, the permissions and current attribute values are checked for consistency with the combined assertions of intent, and enforcement capabilities of the nodes. If they are not consistent, the inconsistencies will be flagged, and the compilation will not produce a policy descriptor. Assertions that depend on values of the attributes are re-checked each time these values change.

3. How Policies are Enforced

3.1. Components of a ZPR Network

The policies ZPL defines are enforced across a group of interconnected networks called a ***ZPRnet***. A ZPRnet consists of a set of communicating ***nodes*** and ***internal services,*** including a ***visa service*** and an ***admin service.*** Nodes move packets toward their destinations and can have ***docks*** that connect to endpoints. Docks allow endpoints to connect to ZPRnet through standard IP networks or by direct connections that use IP protocols.

ZPR defines protocols for node-to-node communication. Nodes send packets to other nodes through ZPR communication ***links*** that may be virtual or physical. Packets can travel between nodes through a sequence of links, called a ***path,*** and policy is enforced by the nodes at each step along the path. For example, a link might bridge one private cloud node to another in the same ZPRnet or carry traffic to a cluster of servers in an enterprise data center.

Each ZPRnet has its own communication policies, and every node of the network enforces those policies on all packets that pass through it. An entire private cloud, real or virtual, might connect to ZPRnet through a single node. Since ZPR does not specify how policy is enforced within a node, the private cloud can use its own internal interfaces, packet formats, and internal network to enforce policy. Docks could be implemented by cloud processors or by specialized network interfaces that serve as gateways for any communication outside of the private cloud's internal network.

3.1.1. Connection to a ZPR Network Using Standard Internet Protocol

Adapters enable IP-based applications to connect docks and communicate through a ZPRnet without modification. They also allow endpoints to communicate with a ZPRnet remotely, through a standard IP network. Adapters can be implemented as operating system extensions or virtual network interfaces that present standard IP interfaces to applications. The adapter communicates with a dock of the ZPRnet through a secure channel called a ***docking session***. The secure channel may communicate through a direct point-to-point connection or a tunnel through a standard IP network. For example, the ZPR Data Protocol (ZDP) can create a secure tunnel through the public internet, allowing remote devices with an adapter to communicate with a ZPRnet.

The adapter and docking session associate the endpoint's communications with the endpoint's authenticated identity and with the identities of any user or services associated with a flow. For example, when a user connects to a ZPRnet remotely, the adapter allows the device to prove its own identity as well as the user's identity to the ZPRnet. The remote device and its adapter do not enforce policy or even know what policies are and they are not trusted components of the ZPRnet, except for their role in authentication. Communication packets travel from a source endpoint through its adapter and docking session to the ZPRnet IP network.

All internal services that support the network are authenticated services within the ZPRnet. All trusted services are also authenticated. These services may be implemented on ZPR endpoints, or they can connect through adaptors and docks.

Adaptors and docking sessions are described in more detail in ZPR-RFC-4. The ZDP protocol is defined in detail in ZPR-RFC-6. Methods of avoiding reliance on trusted third parties in connecting to docks are described in ZPR-RFC-5. An alternate HTTPS Docking Protocol is

described in ZPR-RFC-3. Additional ZPR terminology is defined in ZPR-RFC-2, ZPR-RFC-4, and ZPR-RFC-6.

3.2. Between Nodes, ZPR Operates at the IP Layer

The ZPR communication between nodes occurs at the Internet Layer of the network stack, which roughly corresponds to Layer 3 in the OSI 7-layer model. Because of this, everything that is built on top of IP, including TCP, UDP and all the higher-level protocols that are built on top of these, can work on ZPR without modification. ZPR can run directly on dedicated links, or as an overlay on top of IP. The ZPR configuration mechanism allows links and nodes to enforce policy even during policy updates and reconfiguration of the network.

3.3. Visas

Every packet that travels through a ZPRnet is associated with a visa that verifies that the sender and the receiver are authenticated and that the packet has permission to travel between them. Packets travel between nodes in a format called a ***transit packet.*** Transit packets do not have source or destination addresses in their headers. Instead, they have ***visa identifiers***. They are guided to their destination by the identified ***visas.*** The visa, issued by the visa service, is both a permission to travel and a specification of how the packet is processed as it travels through the network. Visas are unidirectional, so that a visa allowing transmission of a packet from a source to a destination is distinct from the visa that allows transmission of a reply packet from the destination.

The visa identifiers allow very simple enforcement of a packet's compliance with policy, simpler than enforcement of typical IP firewall rules. The translation of policies into network-wide enforcement procedures is done at the time a visa is issued. This simplifies enforcement when packets that reference the visa are being forwarded. A node is not aware of most of the information in the visa. It can determine both policy compliance and the direction to forward a packet from just the visa identifier in the packet's header and the local conditions. This means policy enforcement and forwarding can be done very rapidly, especially on hardware that is optimized for packet processing.

A ***visa*** is a certificate stored in the visa service that has a ***serial number***, an ***expiration time***, a ***visa key*** and ***processing procedures.*** The processing procedures depend on polices and include procedures for forwarding a packet and for encapsulating an IP packet into a transit packet. The encapsulation procedures depend on the header information in the IP packet. The forwarding procedures ensure that a policy-compliant packet will be forwarded along an allowed path and delivered to the appropriate destination. The procedures also ensure that any packet that is not explicitly allowed by policy will be discarded.

On final delivery of a transit packet, the receiver will verify that it was transmitted unaltered. This is enabled by a cryptographic ***message integrity check value (MICV)*** on the payload. The receiver uses a ***visa key*** associated with the visa to verify the MICV. A successful check demonstrates that the message passed through the network unaltered, and that the network correctly maintained the association with the visa and configuration. If the receiver is a dock, it converts the transit packet back to an IP packet and delivers it to the endpoint. If the receiver is a ZPR-node endpoint, the value is checked within the node before it is accepted.

Visas and the methods for distributing visa information to the nodes that need it are described

in more detail in ZPR-RFC-1.

3.4. Compliant Flows

ZPR enforces communications policies by creating a specific type of tamper-proof, secure channel between endpoints called a ***compliant flow***. The ZPR network automatically establishes a compliant flow for any policy-compliant communication between endpoints. Every ZPR packet on a compliant flow has an associated visa that must be valid while the packet transits the network. The packet-processing procedures specified by the visa route the packet to the correct destination and enforce policies as it travels on the flow.

The procedures are applied each time the packet is forwarded, ensuring that the policy enforcement reflects changing conditions and changing attributes. This contrasts with conventional network sessions, which typically enforce policy only at the time a new session is established and at the endpoints or periphery of the network. A compliant flow may have multiple visas associated with it over time, or even at the same time. This allows the flow to outlast the expiration time of an individual visa. Thus, the compliant flow is compliant in both senses of "compliant": it conforms with policy, and it can also adapt to changing conditions, such as the availability of network resources.

All packets in a compliant flow do not necessarily travel across the same path through the network, but ZPR implementations can take advantage of the fact that they often do.

Compliant flows are unidirectional. They are typically used in pairs to create bidirectional flows. Most compliant flows are endpoint-to-endpoint, but ZPR also supports other types of compliant flows such as multicast flows with multiple egress points and ***combining flows*** with multiple ingress points. Combining flows merge packet streams with associative payload-combining functions, such as integer addition or bitwise AND, as they meet each other in transit. Multicast and combining flows can be useful in implementing map/reduce computations that are important in high-performance computing and transport-layer multicast.

Compliant flows are Internet-layer protocols that carry packets on a best-efforts basis, permitting them to be discarded or duplicated. They are defended against tampering, but they have no mechanisms for flow control or retransmission. ZPR uses transport-layer protocols, such as TCP, to establish reliable secure channels between nodes and internal services over compliant flows.

In some cases, policies will route packets through a series of compliant flows. For example, when remote users access a web service, there will typically be some type of load-balancing service gateway between the users and the servers. Packets would travel first over a compliant flow from the user to the service gateway, and then on a different compliant flow from the service gateway to one of the servers. The visas on the first flow would permit the communication between the user and the service gateway, whereas the visas on the second flow would permit communication between the service gateway and the servers. This eliminates the need for a distinct visa for each user/server pair.

Since the nodes and services of a ZPRnet are on the network, compliant flows are not just used in the data plane but also used for carrying the control plane traffic of the ZPRnet. For example, communication between the nodes and the visa service takes place on reliable secure channels established over compliant flows.

The implementation of compliant flows is described in more detail in ZPR-RFC-1 and ZPR-RFC-6.

3.5. Configurations

Policies are considered to be part of a ZPR network's ***configuration***, so changing policies requires reconfiguration. Conventional network reconfiguration often creates security vulnerabilities. ZPR provides a secure method of network reconfiguration that is managed within the ZPR network itself. This helps protect the network from temporary vulnerabilities during reconfiguration, as well as from misconfiguration by careless or malicious administrators.

ZPR allows configurations to be changed while the network continues to operate. Packets will continue to be transported through the network via the policies of the configuration under which they entered the network. This means that multiple configurations may operate concurrently during a changeover.

The ability to handle multiple configurations also allows policies to be tested before they are adopted. When switching to a new configuration, the new configuration status typically progresses from UNDEFINED, to DEFINED, optionally to TESTING, and finally to ACTIVE. Only a single configuration, referred to as the ***active configuration***, is active at any given time. New packets enter the network on the active configuration. When a new configuration becomes ACTIVE, the previously active configuration's status is switched to DEACTIVATING while it continues to operate during the changeover. Once all the deactivating configuration's packets in transit have been delivered, its status reverts to DEFINED.

ZPR's configuration system allows ZPRnets to take advantage of an unconventional method for determining the length of fields in a packet. Most protocols use either fixed packet field lengths or variable-length fields. Fixing field lengths in the protocol definition incurs the inefficiency of covering the entire range of possible use cases, and variable-length fields require extra run-time overhead. ZPR fixes the lengths of fixed fields at configuration time, allowing the packet format to be optimized for each specific use case. The ZPR configuration system supports packets with different formats from multiple configurations traveling through the network concurrently. The only part of a ZPR header that cannot vary between configurations is the field that identifies the configuration that was active when the packet was transmitted.

Configurations and configuration testing are described in more detail in ZPR-RFC-1, ZPR-RFC7 and ZPR-RFC-8.

3.6. ZPR Packets

ZPR networks process two broad types of packets – IP packets that enter the network through a dock, and internal packets that move between nodes across links. IP packets are standard IP packets, in IPv4 or IPv6 format. This section describes the format of internal packets as they are sent over links between network nodes. When a ZPR network is operating as a software-defined overlay network, an internal packet may be encapsulated in a normal IP packet, using the ZPR Data Protocol (ZDP), described elsewhere.

There are two types of internal packets, called ***transit packets*** and ***management packets***, that implement the ZPRnet's data and control planes, respectively. All internal packets specify their type and configuration in the header.

3.6.1. Transit Packets

Transit packets carry data between nodes. A transit packet specifies a ***visa identifier***, a ***payload length*** and, optionally, a ***data attribute tag*** in its header. The visa identifier is a link-specific value that specifies the associated visa. It is local to the link and changes as the packet is forwarded from one link to another, even though the associated visa does not. All information is protected from tampering with a message integrity check value (MICV) that is a cryptographic hash computed from all the information in the packet except the visa identifier, using a cryptographic key associated with the visa.

As illustrated in figure 1, the data payload of the transit packet can carry a compressed representation of an IP-format IP packet. In this case, the IP packet can be reconstructed using combinations of the information in the payload information from the associated visa. More specifically, the fields of the IP packet that have required field values in the packet's visa are not included in the payload and are instead reconstructed at the egress points using information from the packet's associated visa. For example, the source and destination IP addresses of the IP packet are required field values, so they are not included in the payload. Other fields such as protocols, version numbers and ports may be required field values that can be reconstructed. The compressed representation also does not need to include fields, such as an IPv6 hop count field or IPv4 time-to-live (TTL) field, that can be constructed at the egress point.

In many cases, this compression produces transit packets that are smaller than the IP packets they encapsulate. This can be advantageous when the ZPRnet is implemented as a software defined network because it provides the space necessary for the ZPR fields without increasing the packet size, preventing fragmentation of packets at or near the path MTU of the underlying substrate network.

Figure 1 shows an ingress dock converting an IP packet received from an adapter into a transit packet for transmission through a ZPR network and subsequent conversion to an IP packet by an egress dock. The ingress dock receives the IP packet in IP format upon entry into the ZPR network through a docking session from an adapter. The ingress dock converts the IP packet to a transit packet, encoding the information from the IP packet into the payload and the associated visa of the transit packet. The ingress dock uses the payload and visa key to generate the integrity check value through a cryptographic hash.

3.6.2. Management Packets

Management packets are used for control-plane functions such as facilitating the assignment of a local visa identifier for use on the link. They are also used for the distribution or retraction of a visa.

Only the visa service stores the complete information about a visa. Nodes are given only the limited information that they need to know to process packets marked with the visa's identifier. For example, a node will need to know where to forward any packet it processes, but it will not need to know the visa key unless it is an egress node. Management packets are used to distribute this information to where it is needed to process packets and enforce policy.

The visa service, visa identifier assignment and information distribution are described in detail in ZPR-RFC-1, and all types of ZPR packets are described in more detail in ZPR-RFC-6.

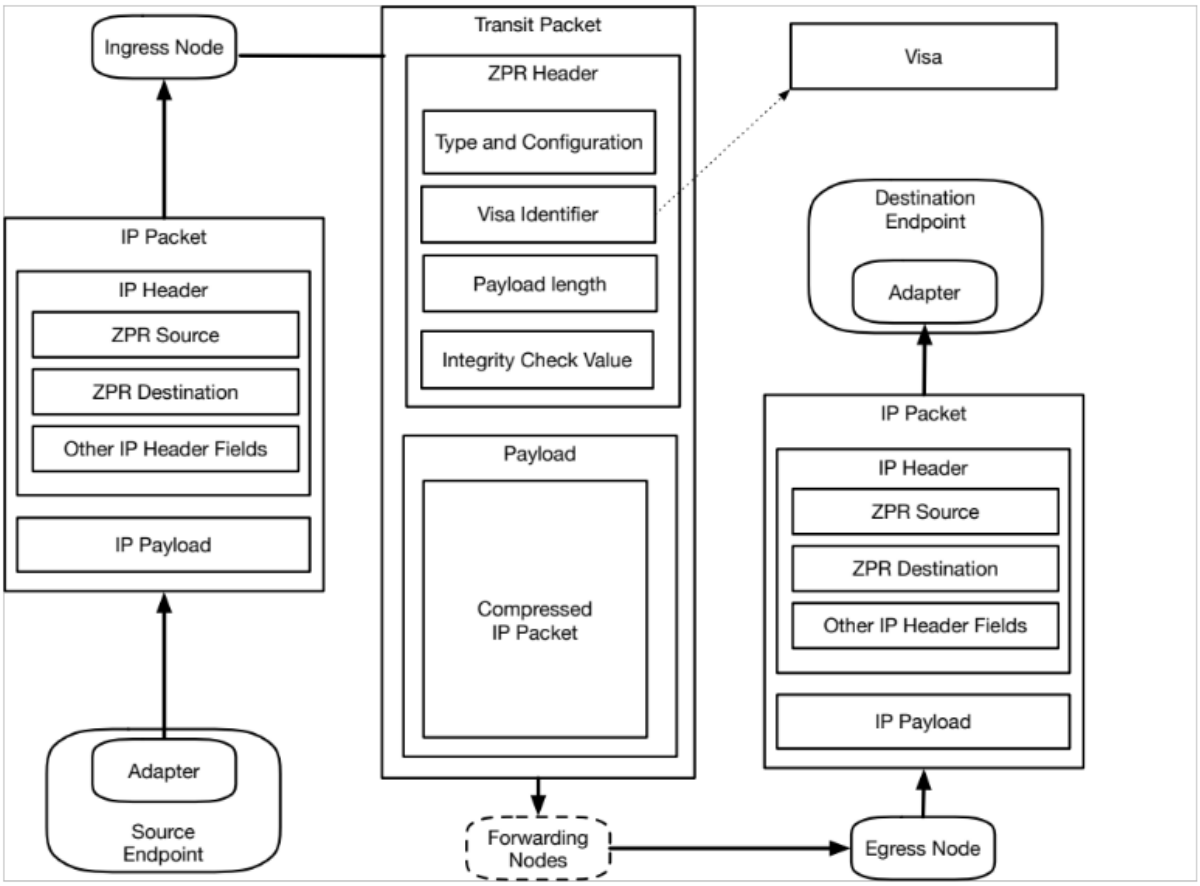

Figure 1: IP packet carried by a transit packet through a ZPRnet.

3.7. ZPR Implemented as a Software Defined Network

Links in a ZPR network may be implemented directly, or they may be implemented virtually on top of an IP substrate network. A combination of direct and virtual links can be used.

When ZPR is implemented with virtual links, as an SDN over a substrate IP network, the substrate may carry other types of traffic. However, there are advantages to using a dedicated substrate network that only carries ZPR traffic. For example, if ZPR is implemented on a dedicated IP network, it is protected against denial-of-service attacks through non-ZPR traffic.

Since policies can restrict the travel of packets to specific sub-networks, a dedicated ZPR-only sub-network may be used to protect particularly sensitive data and essential services. Physically secured areas of a facility and inter-facility communications links can be entirely isolated from non-ZPR traffic.

ZPR implementation of virtual links and software defined network are discussed in more detail in ZPR-RFC-6 and ZPR-RFC-10.

4. ZPR Security

Because a ZPR network has more information than the information within the packets themselves, it can stop attacks that are difficult or impossible to defend against with endpoint protection and firewall partitioning. ZPR can work in addition to these more traditional methods to provide a redundant layer of security.

Part of what distinguishes ZPR from other network security approaches is not just its clearly defined network-enforced policy management, but also its assumptions that bad actors can have access to any part of the network and that they can even compromise network components. ZPR treats all packets with the same level of caution, whether they come from inside or outside the network. The ZPR architecture even assumes that the network administrators can be compromised.

4.1. Paranoid Design Principles

ZPR is designed based on four paranoid design principles:

I. Every packet must have an authenticated sender and receiver.

II. The network itself must enforce communications policies continuously and throughout the network, not just when packets enter and leave or when sessions are established. Every packet must be discarded unless it is explicitly allowed by a policy.

III. The users of the network, the devices that implement the network, and even the individuals who administer the network must not be trusted to always behave in any particular manner. The network must be enforced by policies correctly, even if one of these generally-relied-upon elements becomes unreliable, or even malicious.

IV. The network is configured and managed as a unified resource, through the network itself.

The security value of the last principle may not be obvious, but it is one of the most important security principles of ZPR. By disallowing out-of-band management and forcing all management to be policy compliant, a ZPRnet can protect itself from accidental or deliberate misconfiguration.

For example, it is possible to have a network-enforced policy that requires concurrence of three authenticated administrators from different facilities to activate a new configuration.

ZPR does not depend on out-of-band communication for administration. The secure channels for internal communication are distributed to each node before a configuration becomes active, through the previous configuration. Cryptographic keys are regularly updated through these same secure channels. The only out-of-band communication required is the original startup of a ZPR component, when it is preloaded with cryptographic material that allows it to join the network. Even this step can be protected by requiring a set of keys that is not fully known to any single administrator.

ZPR's limited trust in its own components also limits the damage that can be caused by tampering with a component. Global information about the policies and traffic pattern of ZPRnet is invisible to the individual ZPR nodes, which are only given the specific information they require to process the packets that pass through them. An adapter is trusted only by the specific users and services that use it to communicate. It has no knowledge of network configuration, traffic, connection status or policies at other endpoints. It does not even have knowledge of the policy related to the users and services that connect through it, other than an ability to observe when they can successfully communicate.

The threat resistance of ZPR to bad actors with access is described in ZPR-RFC-9. ZPR's relationship to Zero-Trust is further described in ZPR-RFC-11.

4.2. Byzantine Implementation of Services

Implementations of ZPR networks can use a single visa or admin server, but to fully comply with the ZPR principles, a ZPR network must allow any network component to fail or be compromised without compromising these services. To accomplish this, and to support scalability, these internal services can be implemented by multiple ***servers***, each of which is implemented on a different hardware component. Fault tolerant services can be implemented on redundant servers by known techniques, such as Practical Byzantine Fault Tolerance, or by simpler methods that are effective for the specific functions of these services.

Implementation of Byzantine services is described in ZPR-RFC-1.

5. Summary

Computer networks were originally designed to allow devices to communicate. What we actually need is networks that allow users and services to communicate only when they are allowed to do so. The difference between the two is what makes network security such a hard problem.

ZPR goes to the heart of the problem by providing a way to establish clear policies for communication and providing only communication that is compliant with these policies. ZPR brings identity and policy onto the network stack, adding a robust layer of network security that complements existing security mechanisms.

6. Revision History

1. Revision as of June 12, 2023

    1.1 Explain the problem ZPR solves.

1.2 Reorder sections to describe policy before enforcement mechanisms.

1.3 Delete "network" as example of actor type because it is a confusing special case. The definition remains true as stated because networks can be treated as devices.

1.4 Make a more precise distinction between an attribute and its name and explain type definition.

1.5 Describe ZPR endpoints.

1.6 Postpone explicit mention of the communications system that enables nodes to communicate. This should be in more precise descriptions, but not this overview.

1.7 Describe a private cloud as an example of a node.

1.8 Describe distinction between policy language and complied policy representation.

1.9 Add section on Attributes and trusted services.

1.10 Remove description of visa distribution which is described in other RFCs.

1.11 Figure 1 modified to add data attribute tag and move MICV to header.

2. Revision as of June 20, 2023

2.1 Change title and introduction of section 2.5.

2.2 Rename ZPR endpoints to ZPR-node endpoints to clarify that they exist only within a node.

2.3 Add example in section 3.2.

2.4 Explain how visas simplify enforcement in section 3.3.

2.5 Explain how MICV is checked for ZPR-node endpoints in section 3.

3. Revision as of June 22, 2023

3.1 Point out that identities are primarily used for authentication and logging.

3.2 Explain adapters in more detail in section 3.1.1.

3.3 Retitle section 3.2.

4. Revision as of June 30, 2023

4.1 Update policy language example.

4.2 Explain how visas bind authentication to packets.

4.3 Explain that assertions can specify alerts when unintended patterns of communication are attempted.

5. Revision as of March 8, 2025

5.1 Rename people to users.

5.2 Remove permissions based on type of data transmitted.

5.3 Clarify definition of identity.

5.4 Update Figure 1 to new vocabulary

5.5 Simplify ZPL examples.

5.6 Move configurations to section 3.

5.7 Take out ZPR-node endpoints, as they are distracting.

6. Revision as of May 21, 2025

6.1 Incorporate new endpoint, user, service identity concept.

6.2 Add reference to ZPR-RFC-16.