An Overview of Zero-trust Packet Routing

W. Daniel Hillis, Dave Douglas, Michael Dubno, Frank Kastenholz,
Mathias Kolehmainen, Lew Tucker, Steve Willis, Bruce Walker

Zero-trust Packet Routing (ZPR) creates an identity-aware network security layer, called a ***ZPRnet***, that allows organizations to enforce uniform security policies across all their systems and users. It works seamlessly on-premises, in clouds, and over remote connections. The policies are based on the authenticated identity and attributes of both the communicators and the communicated data. Just as IP enables networks to cooperate in delivering packets, ZPR enables networks to cooperate in enforcing the shared security policies. ZPR is fast and simple enough to implement in either software or hardware. ZPR can use standard IP and requires no changes to existing applications and networks.

This document provides an overview of ZPR and how it works, with a focus on how policies are defined, distributed throughout the network, and enforced each time a packet is forwarded.

How Policies are Defined

The ZPR Policy Language

ZPR policy language, called *ZPL,* is used to specify what packets are allowed to travel over the network (***permissions***), and statements of intent (***assertions***)*,* that are used to verify that permissions are consistent with intent. Policies expressed in other languages can also be complied for a ZPRnet, although other languages may not be able to express all of the types of policies that a ZPRnet can enforce. ZPL is optimized for readability by human policy writers and auditors. ZPL is compiled into a format that can be efficiently enforced in the network.

In ZPR, the senders and receivers that communicate through the ZPRnet are referred to as ***agents***. Agents may be people, devices, software services or combinations of these, such as a person using a specific device, or a service that is implemented by a group of servers. Agents have an authenticated *identity* which has associated *attributes*, such as the role of an employee or serial number of a device. Communication policies in ZPL are stated in terms of attributes of agents, of the data they're communicating, and the circumstances at the time of communication.

All communication on a ZPR network must be specifically allowed by a permission statement. Permissions allow one agent to initiate a communication to a service that is provided by another agent, and they also allow the service to respond. To make the policies easier to understand and audit, all permission statements are positive allowances; ZPL provides no way for one statement to add an exception to another. The advantage of this is that the consequences of each statement can be determined without inspecting all of the other statements.

For example, a permission statement might allow government employees, with the proper clearances, to access classified information on classified database services, connecting through authorized devices, while they are located within a government facility, unless they have exchanged more than 100 megabytes of information in the last 24 hours. In this example, the accessing agents are each a combination of a person and a device. The employment status and clearances are attributes of the people, and the approval status and location are attributes of the device. The amount of data that has been recently communicated is a local circumstance. The limitation on the amount of data transmitted is a condition.

Policies written in ZPL also describe the intent of the combined permissions - what should and should not be allowed. This is accomplished with *assertions.* For example, an assertion might state that no classified information should be communicated to agents outside of a classified facility. Such an assertion does not change the permissions; it specifies a way to automatically check that the permissions are consistent with the intent. A ZPRnet will not allow the adoption of an inconsistent set of policies. If unintended communication is attempted, an assertion can create an alert and optionally 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 attributes of the data communicated, or on global measures of communication, such as limits on the amount of information an agent can move out of a facility during a specific time period.

Policies Depend on Attributes and Circumstances

ZPL allows different types of agents to be defined with required attributes. Attributes are defined by name, type of value, default value, and mutability. The type of value can be a set of elements of a given type. For example, an agent that is an employee might have an attribute named **Employee-ID-number** that has an immutable ID number as a value. An employee might also have an attribute named **mobile-devices** with a mutable value that is a set of mobile devices, such as **{cellphone-6789, laptop-2468, tablet-1357}**. As in this case, elements of the set may have attributes of their own. Typical attributes include type, names, roles, group membership, capabilities, identification numbers and public authentication keys. The definition of agent type may also require a specific value for an attribute.

The *identity* of the agent is an attribute of the agent that is unique, immutable, and authenticatable through one or more authentication services. Each agent has an *authentications* attribute which is the set of currently valid authentications. In the case where the agent is a combination of other agents, such as the combination of a person and device, the identity is the union of the identities of the component agents. For example, an employee connecting through a managed laptop might have an identity with two immutable attributes named **Employee-ID-number** and **Device-serial-number**. Authentication of such a combined identity requires authentication of each of its components.

Identities are primarily used for authentication, logging, and looking up attributes. Policy is normally written in terms attributes, rather than individual identities. While it is possible to write a separate permission for each individual that has access to a company resource, it is more efficient and readable to allow access to all agents that have a specific group attribute.
Circumstances are like attributes in that they have a name and a typed value, but their values are transient and can always be determined by local information. The time, date, and measures of the amount of data recently communicated are examples of circumstances.

Policy Language (ZPL) Examples

A permission for users with an attribute named “role” that has value “customer-support” could be expressed like this:

Allow users with **role customer-support** access to services with **data-category customer-data.**

The permission could also put a limit where the user is located, and the amount of information that they can be transfer per day:

> Allow users with **role customer-support** access to services with **data-category customer-data** when **located-in sales-support-facility,** limited to 100 megabytes-per-day.

An assertion that no vendor should be allowed to access data classified as proprietary could be expressed in the policy language like this:

> Alert and block agents with **personnel-category vendor** access to services with **classification proprietary**.

Assertions may also be made about the allowable values of attributes. An assertion that each device must have at least one manager could be expressed like this:

> There are no devices with **managers {}**.

There are also simple ways to express assertions to limit the number of members in a group or to disallow a person to hold two roles that are incompatible.

Policy For an Organization

The sheer size and complexity of large organizations often requires delegating portions of networks and application management to different groups. For example, local permissions may be added by a local administrator. ZPL supports hierarchies that allow delegation, while still enforcing all assertions.

Since ZPL statements are not permitted to contradict other statements, ZPL policies can be hierarchically combined from several sources to create a ZPRnet’s global policy. To support this, ZPL provides an inclusion mechanism for combining ZPL-format source files maintained by different parts of the organization. The inclusion mechanism allows delegation of policies within specified limits. All permissions are combined and tested against the constraints of assertions higher in the hierarchy, with enterprise-wide policies at the top. Permissions are additive, so they can never conflict with other permissions, but they might conflict with assertions of intent. Such conflicts must be resolved before a new policy is accepted.

The combined policy file includes all the policy permissions and assertions in a form that is convenient to read and audit. The combined file is compiled into a *policy descriptor* which is also human readable but is in a more structured format that is optimized for machine processing. 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 are always checked for consistency with permissions when ZPL is compiled. They can also be run at any time. Assertions should be tested against changes in attribute values, for example when users and groups are modified, when machines are modified, etc. This can be accomplished by running tests on assertions at regular intervals, or by having trusted services report when relevant attributes are changed. Assertions that create alerts are checked whenever new flows of communication are initiated, and they may optionally block the unintended communication.

Trusted Services

The values of attributes come from *trusted services* that provide the value associated with a given agent identity and attribute name. Trusted services may include existing directory services, a trust evaluator, or an identity management system. Values provided by trusted services have expiration times. An API is provided for a service to notify ZPR when values change before their expiration time.

The attributes of communicated data may be determined by attributes of the data source, or by tags on the packets that are transmitted by a trusted data source.
Circumstance restrictions are evaluated locally each time they are used to make a decision.

Configurations

Policies are part of a ZPR network's *configuration*, so changing policies require 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. Multiple configurations may operate concurrently during a changeover. Packets will continue to be transported through the network via the policies of the configuration under which they entered the network.

How Policies Are Enforced

Components of a ZPR Network

A *ZPR network* consists of a set of communicating *nodes* and *internal services,* including a *visa service* and an *admin service.* Nodes can include *docks*, *forwarders* and computers. Docks allow agents to connect to ZPRnet through standard IP networks or by direct connections that use IP protocols. Forwarders move packets between nodes within the ZPR network. Computers can host users or services.

ZPR defines protocols for node-to-node communication. Nodes send packets to other nodes through ZPR communication *links* that may be virtual or physical. Virtual links can be established at configuration time, or dynamically by the visa server. Packets can travel between nodes through a sequence of links, called a *path,* and policy is enforced by the forwarding nodes at each step along the path. Such a link might bridge a private cloud node to another in the same ZPRnet. Another link might extend the ZPRnet to dedicated servers in an enterprise data center.

Each ZPRnet has its own communication policies, and every node of the network enforces those policies. For example, in a cloud where a "smart" network interface card (smartNIC) is attached to a computer that is running multiple computer instances, the smartNIC can act as a node with both dock and forwarder functionality. So, each time a packet passes through a smartNIC it would be checked against policy. Alternatively, a cloud provider might implement an entire cloud tenancy as a single node, by connecting it to the ZPRnet through a gateway that serves the functions of a ZPR adapter and dock.

It will always be possible to write policies that cannot be enforced on a particular network, for example, if there is no trusted source for an attribute that a permission depends upon. A ZPRnet will always verify that it can enforce a set of policies before accepting them. Another example might be a cloud tenancy implemented as a single node that uses micro-segmentation for internal policy enforcement. Such a node might not be able to enforce internal bandwidth limits, so a ZPRnet including that tenancy would not allow policies limiting those bandwidths.

All traffic that implements a ZPRnet flows over ZPR, the visa and administration services and all trusted services are agents within the ZPRnet and communicate over it just like other services.

ZPR Operates at the IP Layer

Communication between ZPR 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.

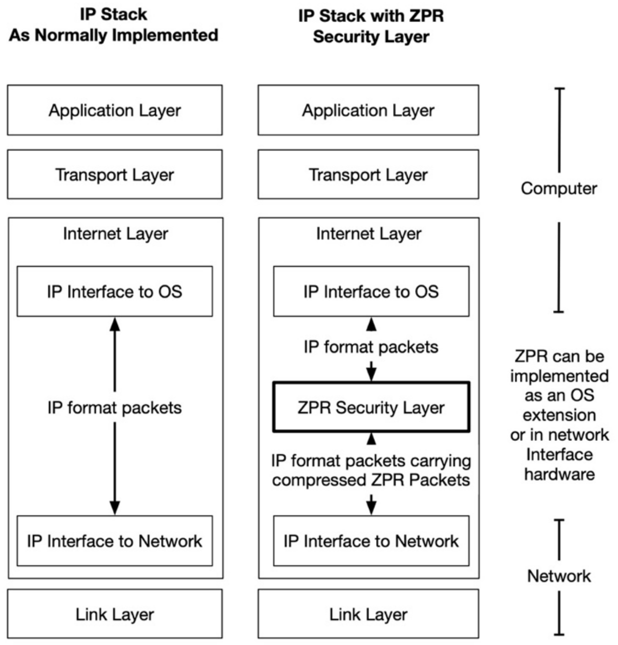

Figure 1. ZPR fits in the IP Stack at the level of the network interface, which may be software or hardware. A standard IP network carries compressed ZPR packets encapsulated in IP packets.

Connecting to a ZPR Network Using Standard Internet Protocol

In ZPR the IP-layer connection between the computer and the network is divided into two parts: the ***adapter*** and the ***dock***. The adapter presents a standard IP interface to the computer and the dock presents a standard IP interface to the network. The dock is a policy-aware component of the ZPRnet, but the adapter is not trusted to enforce policy, nor is it even aware of it.

Adapters can be implemented as operating system extensions, virtual network interfaces, or in a smartNIC between the computer and the network that also implements the dock. An adapter communicates with a dock through a secure communications channel called a *docking session.* The secure channel may be a direct point-to-point connection, a common memory, or a tunnel through a standard IP network. Such a tunnel allows a user to connect remotely to a ZPRnet over the public Internet. The docking session handles both the ZPR agent-to-agent traffic and the management traffic that associates the agent's communications with its authenticated identity. The adapter allows the device to participate in proving its own identity and its user's identity to the ZPRnet.

Visas

One of the security advantages of ZPR is that every node enforces policy without distributing all the policy to the nodes. Most of the work related to determining the policy compliance of a flow of packets is done only when the permission for the flow is granted. That permission is called a *visa*, and it is *issued* by the network's *visa service*. The visa service distributes a small amount of information to every node that will handle the packets traveling under a given visa. The visa identifier in the header of each packet specifies which visa allows and restricts its travel through the network.

When an agent sends a packet to another agent, the packet first goes to the dock where the agent has a docking session. That dock checks if there is an existing visa for the packet, and if not, it requests one from the visa service. The visa service uses the attributes of the sender, receiver, and data, as well as circumstances, to see if policy allows the packet. If so, a visa is issued, and the appropriate information is distributed to the necessary nodes so that the packet will be approved as it traverses the network. If the packet is not allowed by policy, the visa service records the event and notifies the dock to reject the packet.

Visas are end-to-end and allow a specific authenticated agent to transmit packets to another authenticated agent under specific circumstances. These circumstances depend on the time of the communication and may depend on other local state information in the nodes that process the packet. For example, a circumstance may depend on the amount of traffic of a particular class that has recently been transmitted through the node.

Each visa has an expiration time. It may also be *revoked* by the visa service. The permission granted by a visa terminates when the visa expires or is revoked.

Nodes in a ZPRnet are connected by communication links that may by physical or virtual. The visa identifier used for a given visa is established locally on each link, so the identifier for a given visa may vary from link to link. Visa information is pre-distributed along the sequence of nodes that a packet may traverse, and the visa identifiers are established as part of that distribution. The distribution mechanism can also dynamically establish new paths as required.

Distributing Visa Information

Only the visa service stores the entire visa. The visa service distributes information about a visa, but only to the nodes that require it and those nodes are given only the information they need to process packets traveling under the visa. For example:

- In the case of a forwarding node, this information is just the outgoing link to which the packet should be forwarded, the incoming links on which it is allowed to arrive and the local circumstances that need to be checked before forwarding.
- The ingress node of a packet is also given information sufficient for converting an IP packet into its compressed form and a secret key for generating a visa-specific *Message Integrity Check Value* **(***MICV***)**.
- An egress node is given the information required to check the MICV and recreate the original IP packet.

The visa identifiers enable very simple enforcement of a packet's compliance with policy. The translation of policies into network-specific enforcement procedures is done at the time a visa is issued, which simplifies enforcement when packets that reference the visa are being processed. Forwarding a packet only requires checking compliance with local conditions and choosing an outgoing link for the packet. This can be accomplished by simple table lookup, using the visa identifier as an index. The visa identifier allows policy enforcement and forwarding to be implemented very rapidly on a general-purpose computer or on hardware that is optimized for packet processing.

Advantages of Visas

Visas are critical to ZPR security because they bind each packet to its authentication and permissions. The same visa that determines how the packet is forwarded also certifies the authenticated identities of both the sender and receiver as well as the required conditions for travel. The same visa that determines the path also specifies the conditions to be checked at each hop, so the delivery of the packet confirms that it is compliant with policy.

The use of visas has several advantages in ZPR:

1. Binds each packet to the authenticated identity of the sender and receiver, and to the conditions of travel.
2. Enables very efficient packet processing using the visa identifier.
3. Connects the packet to a secret key that allows the receiver to detect if the packet was tampered with or forged.
4. Allow new policies and network topologies to be tested and adopted while a ZPRnet continues to function and enforce policy.
5. Make efficient use of bandwidth since most of the typical IP header packet is not sent in the packet.
6. Enables rapid revocation of a visa, blocking in-transit traffic permissioned by the revoked visa.
7. Enables bandwidth reservation and limits by globally tracking the number and size of messages associated with each class of traffic and distributing local bandwidth constraints to the nodes.

Compliant Flows

ZPR enforces communications policies by creating a specific type of tamper-proof end-to-end secure channel called a *compliant flow.* The ZPR network automatically establishes a compliant flow for any policy-compliant communication between agents. 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. 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. This contrasts with conventional network sessions, which typically enforce policy only when a new session is established. A compliant flow may use multiple visas associated over time, or even at the same time. This allows the flow to outlast the expiration time of an individual visa.

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, so they are typically used in pairs to create bidirectional flows. Most compliant flows are agent-to-agent, but ZPR can also support 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 in high-performance computing and transport-layer multicast.

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 also eliminates the need for a distinct visa for each user/server pair.

Compliant flows are Internet-layer protocols that carry packets on a best-efforts basis, permitting them to discard packets. 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. For example, communication between the nodes and the visa service takes place on reliable secure channels established over compliant flows.

Design Principles

ZPR is designed based on four paranoid design principles:

1. Every packet must have an authenticated sender and receiver.
2. 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.
3. 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.

It should be possible to implement the network so that policies will be enforced correctly, even if one of the network components becomes unreliable or compromised.

4. 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.

An important security advantage of ZPR is that it 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 agents that use it to communicate. It has no knowledge of network configuration, traffic, connection status or policies of other agents. It does not even have knowledge of the policy related to the agents that connect through it, other than an ability to observe when they can successfully communicate.

Summary

Computer networks were originally designed to allow all devices to freely communicate. Security was left to the endpoints and then later augmented by adding other mechanisms, such as firewalls, to stop undesired network traffic. This approach was probably critical for getting internet protocols so widely adopted, but today's organizations need networks that allow users and services to communicate only when they have permission to do so.

ZPR goes to the heart of the problem by providing a way to establish clear communication policies and enforce them with a network that only allows permitted communication. ZPR brings identity and policy down to the level of network packets, adding a robust layer of network security that complements existing security mechanisms.

doi:10.21428/60abf910.5e8e3001

Disclaimer:
This is an overview of Applied Invention's initial proposal for ZPR and does not represent any specific implementation. It is an evolving design and additional versions of this document will be released in the future.