April 2020

EFFICIENT AND SCALABLE PROPAGATION OF SECURITY GROUP MEMBERSHIP IN AN ENTERPRISE WITH RESOLUTION ACROSS MULTIPLE MEMBERSHIP SOURCES

Syam Appala

Rex Fernando

Mike Combs

Rakesh Kandula

Sanjay Hooda

See next page for additional authors

Inventor(s)
Syam Appala, Rex Fernando, Mike Combs, Rakesh Kandula, Sanjay Hooda, and Darrin Miller

EFFICIENT AND SCALABLE PROPAGATION OF SECURITY GROUP MEMBERSHIP IN AN ENTERPRISE WITH RESOLUTION ACROSS MULTIPLE MEMBERSHIP SOURCES

AUTHORS:
Syam Appala
Rex Fernando
Mike Combs
Rakesh Kandula
Sanjay Hooda
Darrin Miller

ABSTRACT

Techniques are described to provide a scalable, secure bindings propagation mechanism. The bindings published by multiple speakers are efficiently reconciled and the filtered messages are notified to the listeners.

DETAILED DESCRIPTION

Security Group Tag (SGT) Exchange Protocol (SXP) version 4 (SXPv4) is a protocol used to propagate Internet Protocol (IP) address - SGT bindings that bind the security group and endpoint IP address together. These IP address - SGT bindings, learnt through SXPv4, are used by Software-Defined Access (SDA) border routers or upstream group-aware firewalls to enforce security group - based access control, security, or experience policies on endpoint flows.

Within SXPv4, there are multiple SXP speakers that create bindings, such as multiple identity services engine instances, the Cloud Policy Connector (CPC), and SXP-capable network devices. These bindings are consumed by SXP listeners, including network devices such as SDA border routers, access switches, security devices (e.g., group-aware firewalls), and applications. SXPv4 uses a peer-to-peer model as the infrastructure for bindings propagation between SXP speakers and listeners. The devices or applications participating in SXP can be a speaker and/or listener, and they may have a separate Transmission Control Protocol (TCP) connection for listening and speaking regarding the bindings.

Each SXP speaker appends its own node identifier (node-id) when sending out IP address - SGT bindings. Each SXP listener checks for its own node-id in the received

update. If it finds its own node-id in the sequence of node-ids, the binding is rejected. Each listener that is also acting as a speaker exports the binding received from another SXP speaker after appending its own node-id. Multiple SXP speakers can create the bindings. The bindings that have been forwarded through the longest set of peers takes precedence.

Figure 1 below illustrates an example network including peer speakers and peer listeners.

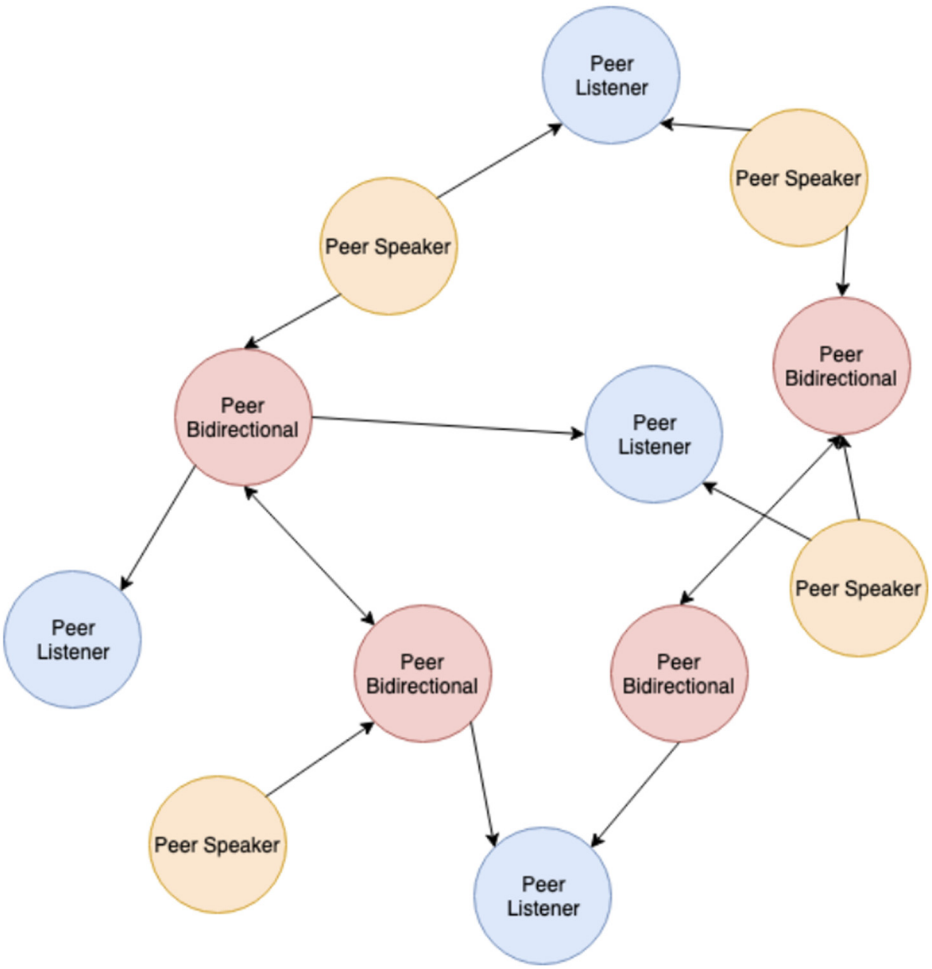

Figure 1

SXPv4 poses a variety of stability, scalability, and security issues for SDA adoption. The connections are proliferated with a peer-to-peer model. The listener has to maintain a separate table for bindings received from multiple speakers, run a binding resolution logic for bindings received from multiple speakers, and create a master list, which is resource-consuming and prohibitive for small footprint devices. Furthermore, every node in SXPv4 requires peer information and a unique key per peer installed for basic MD5 identification.

Also, setting up multiple client peers can be a daunting task. In addition, MD5 verification is a required flag that most kernels do not compile by default, making peer integrations on the application layer impossible in most cases. The security of the connection is also primitive (MD5).

SXPv4 also creates binding conflicts wherein two or more speakers attempt to create the same binding with different secondary tag designations, and one notification can override the other. Binding forwarding loops can also be created if a peer forwards tags created across the same chain of peers. SXPv4 may be unable to achieve the desired tag precedence because of the way the bindings are propagated and precedence is established. Moreover, the network device has to maintain one connection per Virtual Network (VN), leading to a connection explosion. Consider a scenario with sixty fabrics having two border routers per fabric and ten VNs per fabric – this yield 1200 connections. Also, SXP supports domains, which are essentially collections of SXP peers that can receive the bindings. There is no flexibility with regard to message filtering. In addition, the SXP data model is fixed and not extensible. There are many requirements from the field (e.g., to add secondary tags to the primary tag) and from projects such as Multi-Domain Policy (MDP).

These problems arise from the architecture of SXPv4 itself. Moving from a peer-to-peer model to a secure, scalable hub-and-spoke model addresses the security and connection proliferation problems and makes tag resolution easier to reason, prevents infinite binding cycles, and creates a single source of truth for the state of group tags. Listeners can now directly subscribe only to the VNs / Virtual Route Forwarding (VRF) nodes that they are interested in rather than the entire network, in a single connection. With the message broker - based hub-and-spoke model described herein, the participants can share an extensible data model to include secondary tags with the primary tags, and may also share the full qualification(s) of the SGT (e.g., controller, tenant, VN, SGT, etc.).

New listeners may immediately require all existing tags per VN interest. In this model they can directly take a snapshot of the bindings table (per VN) and consume it immediately in its entirety.

No setup is required other than pointing the peers to a message broker.

Figure 2 below illustrates an example speaker-to-listener workflow.

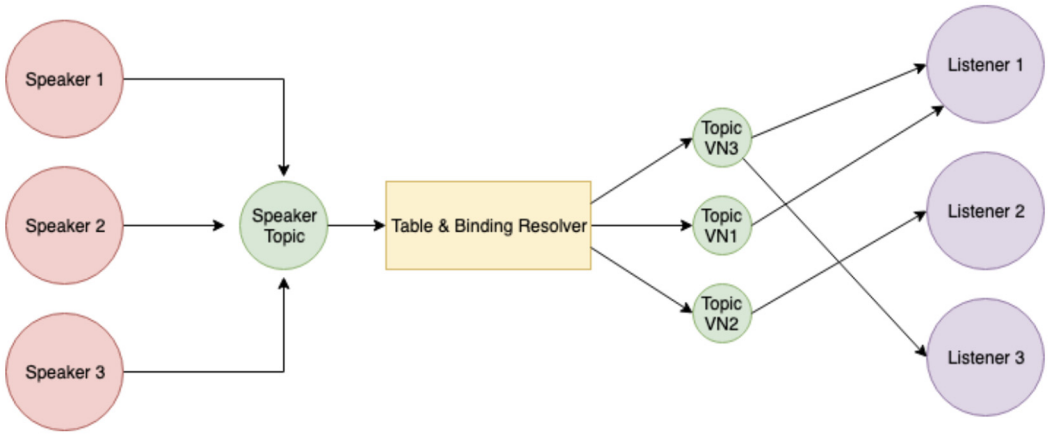

Figure 2

The speaker flow begins when the speaker connects to the system and the discovery process is initiated. The speaker declares that it will create bindings for VNs x and y. If those VNs did not previously exist, the system may automatically create those VN-specific topics (topic x and topic y) and create an in-memory table. The speaker is given a speaker topic to begin writing, and the speaker can now publish the IP address - SGT bindings on the speaker topic.

The listener flow begins when a listener connects to a system and the discovery process begins. The listener declares that it requires bindings from VNs x and y. Topic x has had bindings written to it previously, whereas topic y has not. As such, the broker prepares a table synchronization based on VN x and writes to topic x' created for the purpose of table synchronization with the listener. The broker sends topic y to the listener so that the listener can subscribe. The broker dumps the entire state snapshot to topic x', sends the topic x' to the listener along with a number of messages, and ends with an offset of the last message written to topic x. The listener consumes messages from the topic x' up to the given offset, sends a synchronization acknowledgement to the broker, unsubscribes from topic x', and subscribes to topic x from the offset sent by the broker. The broker then deletes topic x'.

A binding resolution was previously established in SXPv4 by a priority system based on the size of the chain of peers that had propagated that binding. A different mechanism is provided herein. First, the broker receives a binding from a speaker subscribed to VN x. The broker checks the table to determine whether the binding (unique

key IP address and primary group tag) already exists. If so, the broker creates "diffs" and writes the new binding to VN topic x if it is different. The speaker that created the duplicate binding has an edge pointer created that points to the tag (and secondary groups, if applicable). If a speaker attempts to delete the binding, but the tag includes other speakers that have also declared that binding, only its edge is removed. If there are no more edges for a binding, that binding is deleted and a delete message is sent to the VN topic. If a speaker that has deleted its edge to a binding has added unique secondary groups, those groups are removed and an update binding is sent to the VN topics.

The hub-and-spoke model described herein alleviates the connection explosion problem by drifting the bindings propagation from a connection-centric approach to message processing - centric approach. Listeners simply have to establish one connection with the broker, and the broker performs the operations necessary to propagate reconciled, filtered bindings to the listeners. This extensible data model may include more attributes such as secondary tags and full SGT qualification(s) (e.g., controller, tenant, VN constructs, etc.). There may also be an efficient bindings resolution if multiple sources are publishing the same bindings, whereby the broker suppresses the duplicate bindings and retains and propagates the more complete secondary tag information associated with the bindings to the listeners.

The broker may discover the VNs of interest from the listeners, create VN-specific topics and topic membership, apply VN filters on the bindings, and propagate the filtered bindings on the VN topics. The broker may also manage the lifecycle of the bindings and topics and purge them when there are zero speakers associated with the bindings or when there are zero listeners participating in VN-specific topics. Moreover, listener-broker-speaker connections are secured by Transport Layer Security (TLS) based mutual authentication and encryption instead of the MD5 signature used by SXPv4 today. Scaling of the deployment is a function of broker scale and its ability to support greater volumes of bindings exchange and participants. These techniques may support on-premise or cloud deployments, and may avoid firewall pinholing depending on the broker deployment.

As noted, there are issues with the current SXP protocol and the associated limitations in SDA. The techniques described herein provide a scalable mechanism to propagate the bindings using a publish-subscribe broker. In addition to providing a scalable,

highly available, reconciled, multiple-source binding propagation to the listeners (e.g., network devices, applications deployed on-premise or via the cloud, etc.), a method is also provided for the broker to create VN-specific topics for the bindings speaker and manage the topics through the lifecycle automatically. The broker purges them when there are zero speakers associated with the bindings or when there are zero listeners participating in VN-specific topics.

The broker may also map listeners' VN interests to the specific VN topics created and have the listeners subscribe to the topics. The broker may also discover preexisting, published bindings on a given VN-specific topic (e.g., VN1 topic), create a temporary topic (e.g., VN1' topic) to provide a bulk update of those bindings to the newly connected listener, delete the temporary topic (VN' topic) after the bulk update, and subscribe the listener to the main VN topic (VN1) in order for the listener to get real-time updates. Bindings may be efficiently resolved if multiple sources publish the same bindings. The broker may suppress the duplicate bindings, and retain and propagate the more complete secondary tag information associated with the bindings to the listeners. In the case of SDA, in addition to "SDA transit," "IP transit," which does not carry SGT tags across sites, is also supported. The bindings can be propagated without scale, availability, and reconciliation concerns in a data transport agnostic manner. This is a useful, prudent way to address the aforementioned problems and drive SDA adoption.

In summary, techniques are described to provide a scalable, secure bindings propagation mechanism. The bindings published by multiple speakers are efficiently reconciled and the filtered messages are notified to the listeners.