Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 22, 2017

M. Kuehlewind
B. Trammell
ETH Zurich
J. Hildebrand

October 19, 2016

Transport-Independent Path Layer State Management
draft-trammell-plus-statefulness-00

Abstract

This document describes a simple state machine for stateful network devices on a path between two endpoints to associate state with traffic traversing them on a per-flow basis, as well as abstract signaling mechanisms for driving the state machine. This state machine is intended to replace the de-facto use of the TCP state machine or incomplete forms thereof by stateful network devices in a transport-independent way, while still allowing for fast state timeout of non-established or undesirable flows.

Table of Contents

1. Introduction

The boundary between the network and transport layers was originally defined to be that between information used (and potentially modified) hop-by-hop, and that used end-to-end. End-to-end information in the transport layer is associated with state at the endpoints, but processing of network-layer information was assumed to be stateless.

The widespread deployment of stateful middleboxes in the Internet, such as network address and port translators (NAPT), firewalls that model the TCP state machine to distinguish packets belonging from desirable flows from backscatter and random attack traffic, and devices which keep per-flow state for reporting and monitoring purposes (e.g. IPFIX [RFC7011] Metering Processes), has broken this assumption, and made it more difficult to deploy non-TCP transport protocols in the Internet.

The deployment of new transport protocols encapsulated in UDP with encrypted transport headers (such as QUIC [I-D.hamilton-quic-transport-protocol]) will present a challenge to the operation of these devices, and their ubquity likewise threatens to impair the deployability of these protocols. There are two main causes for this problem: first, stateful devices often use an internal model of the TCP state machine to determine when TCP flows start and end, allowing them to manage state for these flows; for UDP flows, they must rely on timeouts. These timeouts are generally short relative to those for TCP [IMC-GATEWAYS], requiring UDP-encapsulated transports either to generate unproductive keepalive traffic for long-lived sessions, or to tolerate connectivity problems and the necessity of reconnection due to loss of on-path state.

This document presents a solution to this problem by defining a state machine that is transport independent to be implemented at per-flow state-keeping middleboxes as a replacement for incomplete TCP state modeling. Middleboxes implementing this state machine using signals from a common UDP encapsulation layer can have equivalent necessary state information to that provided by TCP, reducing the friction between middleboxes and these new transport protocols.

2. Terminology

In this document, the term "flow" is defined to be compatible with the definition given in [RFC7011]: A flow is defined as a set of packets passing a device on the network during a certain time interval. All packets belonging to a particular Flow have a set of common properties. Each property is defined as the result of applying a function to the values of:

1. one or more network layer header fields (e.g., destination IP address) or transport layer header fields (e.g., destination port number) that the device has access to;

2. one or more characteristics of the packet itself (e.g., number of MPLS labels, etc.);

3. one or more of the fields derived from packet treatment at the device (e.g., next-hop IP address, the output interface, etc.).

A packet is defined as belonging to a flow if it completely satisfies all the defined properties of the flow.

A bidirectional flow or biflow is defined as compatible with [RFC5103], by joining the "forward direction" flow with the "reverse direction" flow, derived by reversing the direction of directional fields (ports and IP addresses). Biflows are only relevant at

devices positioned so as to see all the packets in both directions of the biflow, generally on the endpoint side of the service demarcation point for either endpoint as defined in the reference path given in [RFC7398].

3. State Machine

The transport-independent state machine for on-path devices is shown in Figure 1. It relies on four states, three configurable timeouts, and a set of signals defined in Section 4. The states are defined as follows:

- zero: there is no state for a given flow at the device
- uniflow: a packet has been seen in one direction; state will be kept at the device until it is explicitly cancelled or until timeout t1 elapses without a packet.
- biflow: a packet has been seen in one direction and an indication that that the receiving endpoint wishes to continue the association has been seen in the opposite direction; state will be kept at the device until it is explicitly canceled or until timeout t2 elapses without a packet.
- closing: an established biflow is shutting down due to an explicit close indication; state will be kept at the device until timeout t3 elapses.

The three timeouts are defined as follows:

- t1 is the unidirectional idle timeout. It can be considered equivalent to the idle timeout for transport protocols where the device has no information about session start and end (e.g. most UDP protocols).
- t2 is the bidirectional idle timeout. It can be considered equivalent to the timeout for transport protocols where the device has information about session start and end (e.g. TCP).
- t3 is the closing timeout: how long the device will account additional packets to a flow after observing a stop signal, and is generally applied to ensuring a reordered stop signal doesn't create a new flow.

Selection of timeouts is a configuration and implementation detail, but generally t3 <= t1 < t2.

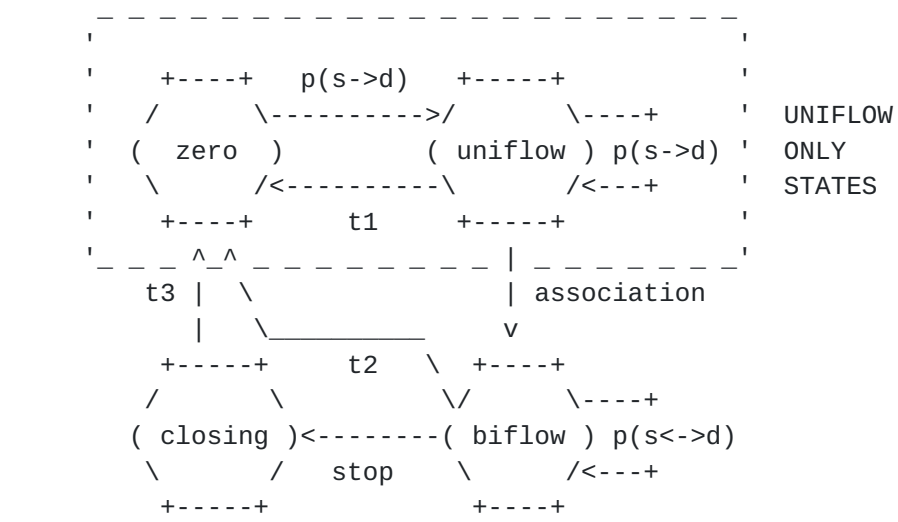

Figure 1: Transport-Independent State Machine for Stateful On-Path Devices

3.1. Uniflow States

Every packet received by a device keeping per-flow state must associate that packet with a flow (see Section 4.1). When a device receives a packet associated with a flow it has not state forward, and it is configured to forward the packet instead of dropping it, it moves that flow from the zero state into the uniflow state and starts a timer t1. It resets this timer for any additional packet it forwards in the same direction as long as the flow remains in the uniflow state. When timer t1 expires on a flow in the uniflow state, the device drops state for the flow and performs any processing associated with doing so: tearing down NAT bindings, closing associated firewall pinholes, exporting flow information, and so on. The device may also drop state on a stop signal, if observed.

Some devices will only see one side of a communication, e.g. if they are placed in a portion of a network with asymmetric routing. These devices use only the zero and uniflow states (as marked in Figure 1.) In addition, true uniflows - protocols which are solely unidirectional (e.g. some applications over UDP) - will also use only the uniflow-only states. In either case, current devices generally don't associate much state with observed uniflows flows, and timeout is generally sufficient to expire this state.

3.2. Biflow States

A uniflow transitions to the biflow state when the device observes an association signal. An association signal consists of a packet sent in the opposite direction from the uniflow packet(s), with certain properties as defined in Section 4.2. After transitioning to the biflow state, the device starts a timer t2. It resets this timer for any packet it forwards in either direction. The biflow state represents a fully established bidirectional communication. When timer t2 expires, the device assumes that the flow has shut down without signaling as such, and drops state for the flow, performing any associated processing. When a stop signal is observed in either direction, the flow transitions to the closing state.

When a flow enters the closing state, it starts a timer t3. While the stop signal should be the last packet on a flow, the t3 timer ensures that reordered packets after the stop signal will be accounted to the flow. When this timer expires, the device drops state for the flow, performing any associated processing.

3.3. Additional States and Actions

This document is concerned only with states and transitions common to transport- and function- independent state maintenance. Devices may augment the transitions in this state diagram depending on their function. For example, a firewall that decides based on some information beyond the signals used by this state machine to shut down a flow may transition it directly to a blacklist state on shutdown. Or, a firewall may fail to forward additional packets in the uniflow state until an association signal is observed.

4. Abstract Signaling Mechanisms

The state machine in Section 3 requires three signals: a new flow signal (the first packet observed in a flow in the zero state), an association signal (allowing a device to verify that an endpoint wishes a bidirectional communication to be established or to continue), and a stop signal (noting that an endpoint wishes to stop a bidirectional communication). Additional related signals may also be useful, depending on the function a device provides. There are a few different ways to implement these signals; here, we explore the properties of some potential implementations.

We assume the following general requirements for these signals; parallel to those given in [draft-trammell-plus-abstract-mech]:

- At least the endpoints can verify the integrity of the signals exposed, and shut down a transport association when that

verification fails, in order to reduce the incentive for on-path devices to attempt to spoof these signals.

- Endpoints and devices on path can probabilistically verify that a originator of a signal is on-path.

4.1. Flow Identification

In order to keep per-flow state, each device using this state machine must have a function it can apply to each packet to be able to extract common properties to identify the flow it is associated with. In general, the set of properties used for flow identification on presently deployed devices includes the source and destination IP address, the source and destination transport layer port number, the transport protocol number. The differentiated services field [RFC2474] may also be included in the set of properties defining a flow, since it may indicate different forwarding treatment.

However, other protocols may use additional bits in their own headers for flow identification. In any case, a protocol implementing signaling for this state machine must specify the function used for flow identification.

4.2. Association Signaling

An association signal indicates that endpoint that received the first packet seen by the device is interested in continuing conversation with the sending endpoint. This signal is roughly an in-band analogue to consent signaling in ICE [RFC7675] that is carried to every device along the path.

Transport-independent, path-verifiable association signaling can be implemented using a association token generated by one endpoint, present on each packet sent in the flow by that endpoint, and a response token, derived from the association token using a well-known, defined function, present on each packet sent in the flow by the opposite endpoint. We can assume a transport association has an initiator and a responder; under this assumption, the association token is chosen by the initiator, and the response token generated by the responder.

Any packet sent by the responder with a valid response token, and without a stop signal (see Section 4.3), can then be taken to be a association signal to continue a bidirectional communication. Note that, since it relies on a widely-known function, this mechanism does allow on-path devices to forge association signaling in a way that downstream on-path devices cannot detect. However, in the presence of end-to-end signal integrity verification, this forgery will be

detected by the endpoint, which MUST terminate the association on a forged association signal; the flow at the duped on-path device will transition from biflow to closing within a single packet. This reduces any attack against the association signaling mechanism to the disruption of a connection, which on-path devices can do in any case by simply refusing to forward packets.

Association tokens MUST be chosen by initiators to be hard to guess; otherwise, off-path devices can spoof association and response signals. Cryptographic random number generators suffice here. In choosing the number of bits for an association token, there is a tradeoff between per-packet overhead and state overhead at on-path devices, and assurance that an association token is hard to guess. This tradeoff must be evaluated at protocol design time.

There are a few considerations in choosing a function (or functions) to generate the response token from the association token, and to verify a response token given an association token. The simplest such function is the identity function: the response token is simply the association token. Simple two-way functions (e.g. one's complement of the association token) provide additional assurance of implementation of the protocol, and cannot be accidentally triggered by simple reflection of unknown bits in a packet. One- way functions (e.g. truncated SHA-2 hash of the association token) additionally allow on-path recognition of initiator and responder from the middle of a flow.

In any case, a concrete implementation of association signaling must choose a single function, or mechanism for unambiguously choosing one, at both endpoints as well as along the path.

4.3. Stop Signaling

A stop signal is directly carried or otherwise encoded in the protocol header to indicate that a flow is ending, whether normally or abnormally, and that state associated with the flow should be torn down. Upon decoding a stop signal, a device on path should move the flow from uniflow state to null, or from biflow state to closing.

Transports should send a stop signal only on the last packet sent in a bidirectional flow. The closing timeout t3 is intended to ensure that any packets reordered in delivery are accounted to the flow before state for it is dropped.

We assume the encoding of a stop signal into a packet header, as with all other signals, is integrity protected end-to-end. Stop signals, as association signals, be forged by one on-path device to dupe other devices into moving flows into the closing state. However, state

will be re-established by the continuing flow (and resulting association signals) after the closing timeout, and an endpoint receiving a spoofed stop signal could enter a fast re-establishment phase of the upper layer transport protocol to minimize disruption, further reducing the incentive to attackers to spoof stop signals.

4.4. Timeout Exposure

Since one of the goals of these mechanisms is to reduce the amount of non- productive keepalive traffic required for UDP-encapsulated transport protocols, they MAY be deployed together with a path-to-receiver signal with feedback as defined in [draft-trammell-plus-abstract-mech] asking for timeouts t1, t2, and t3 for a given flow.

5. Signal mappings for transport protocols

We now show how this state machine can be driven by signals available in TCP and QUIC.

5.1. Signal mapping for TCP

A mapping of TCP flags to transitions in to the state machine in Section 3 shows how devices currently using a model of the TCP state machine can be converted to use this state machine.

Simply speaking, an ACK flag [RFC0793] in the absence of the FIN or RST flags, and an in-window acknowledgment number, is synonymous with the association signal, while the RST or final FIN flags are stop signals. For a typical TCP flow:

1. The initial SYN places the flow into uniflow state,

2. The SYN-ACK sent in reply acts as a association signal and places the flow into biflow state,

3. Any RST moves the flow into closing state, or

4. The final FIN-ACK (not the first half-close FIN) moves the flow into closing state.

Note that since any valid ACK acts as an association signal, this mapping allows flows to transition to the biflow state even if the initial SYN-ACK is not observed. However, generating a stop signal from FIN does require additional TCP state modeling to prevent moving into the closing state on a half-close.

Note that the association and stop signals derived from the TCP header are not integrity protected, and an association signal based on in-window ACK is not particularly resistant to off-path attacks; the state machine is therefore more susceptible to manipulation when used with vanilla TCP as when with a transport protocol providing full integrity protection for its headers end-to-end.

5.2. Signal mapping for QUIC

QUIC [I-D.hamilton-quic-transport-protocol] as presently defined lacks only a stop signal to be able to drive this state machine. QUIC's 64-bit connection ID suffices as an association and response token as in Section 4.2; the response token function is identity. QUIC's cryptographic protocol, to be based on TLS [I-D.thomson-quic-tls], will provide the necessary integrity protection to drive the state machine.

Any number of designs could be chosen to add a stop signal compatible with the definition in Section 4.3 to QUIC. One is particularly promising, however. We note that the Public Reset facility described in section 8 of [I-D.hamilton-quic-transport-protocol] very nearly meets the criteria; it would need to be expanded to expose normal termination as well as abnormal termination, and to provide for endpoint detection of inauthentic termination signals.

6. IANA Considerations

This document has no actions for IANA.

7. Security Considerations

This document defines a state machine for transport-independent state management on middleboxes, using in-band signaling, to replace the commonly- implemented current practice of incomplete TCP state modeling on these devices. It defines new signals for state management. While these signals can be spoofed by any device on path that observes traffic in both directions, we presume the presence of end-to-end integrity protection of these signals provided by the upper-layer transport driving them. This allows such spoofing to be detected and countered by endpoints, reducing the threat from on-path devices to connection disruption, which such devices are trivially placed to perform in any case.

8. Acknowledgments

Thanks to Christian Huitema for discussions leading to this document.

This work is partially supported by the European Commission under Horizon 2020 grant agreement no. 688421 Measurement and Architecture for a Middleboxed Internet (MAMI), and by the Swiss State Secretariat for Education, Research, and Innovation under contract no. 15.0268. This support does not imply endorsement.

9. References

9.1. Normative References

[RFC5103] Trammell, B. and E. Boschi, "Bidirectional Flow Export Using IP Flow Information Export (IPFIX)", RFC 5103, DOI 10.17487/RFC5103, January 2008, <http://www.rfc-editor.org/info/rfc5103>.

[RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <http://www.rfc-editor.org/info/rfc7011>.

[RFC7398] Bagnulo, M., Burbridge, T., Crawford, S., Eardley, P., and A. Morton, "A Reference Path and Measurement Points for Large-Scale Measurement of Broadband Performance", RFC 7398, DOI 10.17487/RFC7398, February 2015, <http://www.rfc-editor.org/info/rfc7398>.

9.2. Informative References

[draft-trammell-plus-abstract-mech] Trammell, B., "Abstract Mechanisms for a Cooperative Path Layer under Endpoint Control", September 2016.

[I-D.hamilton-quic-transport-protocol] Hamilton, R., Iyengar, J., Swett, I., and A. Wilk, "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-hamilton-quic-transport-protocol-00 (work in progress), July 2016.

[I-D.thomson-quic-tls] Thomson, M. and R. Hamilton, "Porting QUIC to Transport Layer Security (TLS)", draft-thomson-quic-tls-00 (work in progress), March 2016.

[IMC-GATEWAYS] Hatonen, S., Nyrhinen, A., Eggert, L., Strowes, S., Sarolahti, P., and M. Kojo, "An experimental study of home gateway characteristics (Proc. ACM IMC 2010)", n.d..

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <http://www.rfc-editor.org/info/rfc793>.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <http://www.rfc-editor.org/info/rfc2474>.

[RFC7675] Perumal, M., Wing, D., Ravindranath, R., Reddy, T., and M. Thomson, "Session Traversal Utilities for NAT (STUN) Usage for Consent Freshness", RFC 7675, DOI 10.17487/RFC7675, October 2015, <http://www.rfc-editor.org/info/rfc7675>.

Authors' Addresses

Mirja Kuehlewind
ETH Zurich
Gloriastrasse 35
8092 Zurich
Switzerland

Email: mirja.kuehlewind@tik.ee.ethz.ch

Brian Trammell
ETH Zurich
Gloriastrasse 35
8092 Zurich
Switzerland

Email: ietf@trammell.ch

Joe Hildebrand

Email: hildjj@cursive.net