PLUS BoF
Internet-Draft
Intended status: Informational
Expires: April 1, 2017

B. Trammell
ETH Zurich

September 28, 2016

Abstract Mechanisms for a Cooperative Path Layer under Endpoint Control

draft-trammell-plus-abstract-mech-00

Abstract

This document describes the operation of three abstract mechanisms for supporting an explicitly cooperative path layer in the Internet architecture. Three mechanisms are described: sender to path signaling with receiver integrity verification; path to receiver signaling with confidential feedback to sender; and direct path to sender signaling.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 1, 2017.

Copyright Notice

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 network address and port translation (NAPT) in the Internet has eroded this boundary. Since the first four bytes after the IP header or header chain - the source and destination ports - are frequently used for forwarding and access control decisions, and are routinely modified on path, they have de facto become part of the network layer. In-network functions that exploit the fact that transport headers are in cleartext in the absence of widespread deployment of IPsec [RFC4301] further erode this boundary.

Evolution above the network layer and integrity of transport layer functions is only possible if this layer boundary is reinforced. Asking on-path devices nicely not to muck about in the transport layer and below - stating in an RFC that devices on path MUST NOT use or modify some header field - has not proven to be of much use here. A new approach is necessary, consisting of cryptographic integrity

protection of network and transport layer headers which the endpoints choose to expose to the path, and cryptographic confidentiality protection of transport layer headers not to be exposed.

We define the "path layer" to consist of the headers and associated functions that are explicitly exposed to devices along the path in this scheme. This document describes three abstract mechanisms for implementing path layer communications. The first principles in the design of these mechanisms are endpoint control, signaling transparency, and support of arbitrary relationships between endpoints and path devices.

The principle of endpoint control means that all signaling, even from the path, is initiated by a sending endpoint, allowing a sending endpoint to opt into or out of path layer communications as it sees fit.

The principle of signaling transparency means that at least the semantic type of all signals using these mechanisms must be visible to all path elements. This makes it possible for users and network operators to use traffic inspection to observe what is being signaled. As a last resort, users and networks wishing to limit signaling using these mechanisms can simply drop packets containing signals they would prefer not to have sent.

The principle of arbitrary relationship means that the basic mechanisms do not require any trust or cryptographic state between endpoints and path elements to function, though integrity protection and confidentiality for communication with path elements can be layered over these mechanisms; definition of key exchange and cryptographic protocols for this layering is out of scope for this document, however.

2. Mechanism Definitions

Three abstract mechanisms suffice to implement in-band path layer communications, given our first principles. First, a sender can make declarations about itself or about traffic it is sending to devices along the path, relying on the receiver to verify integrity of the declaration. Second, a sender can allow path elements to make declarations about themselves or their treatment of given traffic, by creating space for the path elements to do so. In this case, the integrity of the presence, type, and size of this space is verified by the receiver, but not the content of the declaration made by the path. Third, a path element can make a declaration about a dropped packet back to the sender of that packet.

These mechanisms are described in an implementation-independent way; however, there are a few basic assumptions made by the design:

- There exists a technique by which packets can be selected and grouped by the sending endpoint such that these groups are visible to devices along the path (e.g. using an N-tuple).
- The mechanisms can add information to selected packets in a communication between two endpoints.
- The mechanisms use a shared secret between the two endpoints provided by some upper layer.
- The confidentiality and integrity of upper layer's headers and payload are cryptographically protected by the upper layer.
- The declarations carried by the mechanisms can be expressed in terms of key- value pairs, such that the type and semantic meaning of the declaration are completely defined by the key. The mechanisms don't necessarily need to be implemented using a generic key-value framing (e.g. CBOR [RFC7049]); the key can be implied by a position in a defined packet header.

2.1. Sender-to-Path Declarations

To make a declaration about a packet or flow to all path elements, the sender adds a key-value pair to a packet within the flow. The fact that this is a sender-to-path declaration is part of the definition of the key. Multiple declarations can appear in a single packet. All the declarations within a packet, together with other transport and network layer information which must not be modified by the path, are protected by a message authentication code (MAC) sent along with the packet, generated with a key derived from a secret known only to the endpoints.

The receiver then verifies the MAC on receipt. Verification failure implies an attempt to modify the header used by this mechanism, and therefore must cause the transport association to reset.

This arrangement is illustrated in Figure 1.

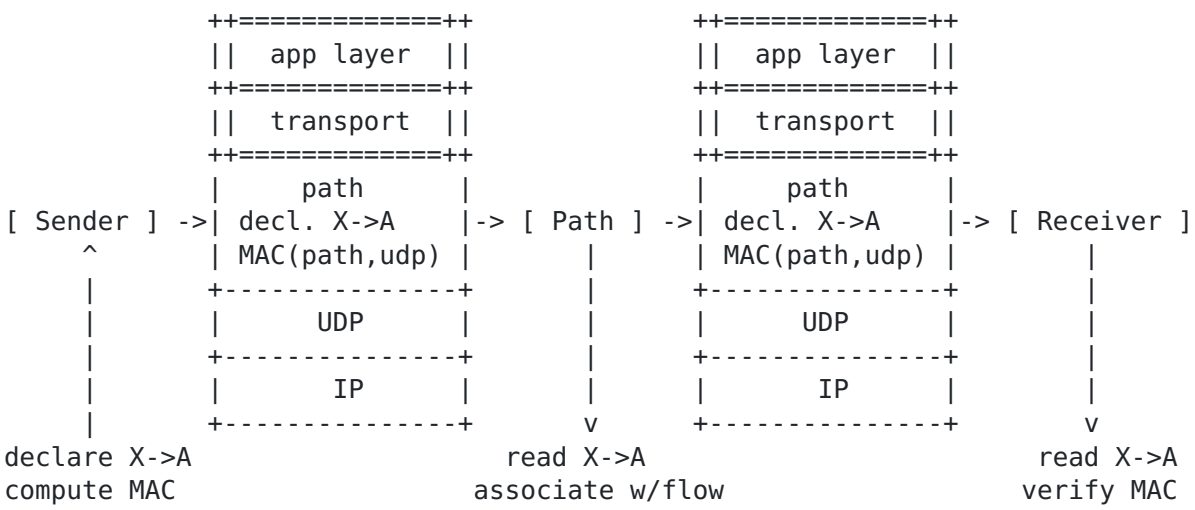

Figure 1: Sender to Path Declaration

2.2. Path-to-Receiver Declarations with Feedback

To allow one or more path elements to make a declaration about itself with respect to a packet or a flow to the receiver, the sender adds a key-value pair to a packet within the flow. The fact that this is a path-to-receiver declaration is part of the definition of the key. Further, the value has a fixed length of N bytes (which my also be part of the definition of the key). Path-to-receiver declarations may be combined in a packet with sender-to-path declarations as in Section 2.1, and are covered by the same MAC. However, when calculating the MAC for a path-to-receiver declaration, its value is assumed to be an N-byte array of zeroes. The MAC therefore protects the presence of the key and the length of the value, but not its content.

The initial value of a path-to-receiver declaration is up to the sender, and is generally defined by the declaration itself. The behavior of a path element in filling in a path-to-receiver declaration given which value is already present is also part of the declaration defintion. Declarations may accumulate by some operation (e.g., max, min, sum for measurement declarations), be determined by the first or last path element, or be addressed to a specific path element to fill in.

This arrangement is illustrated in Figure 2.

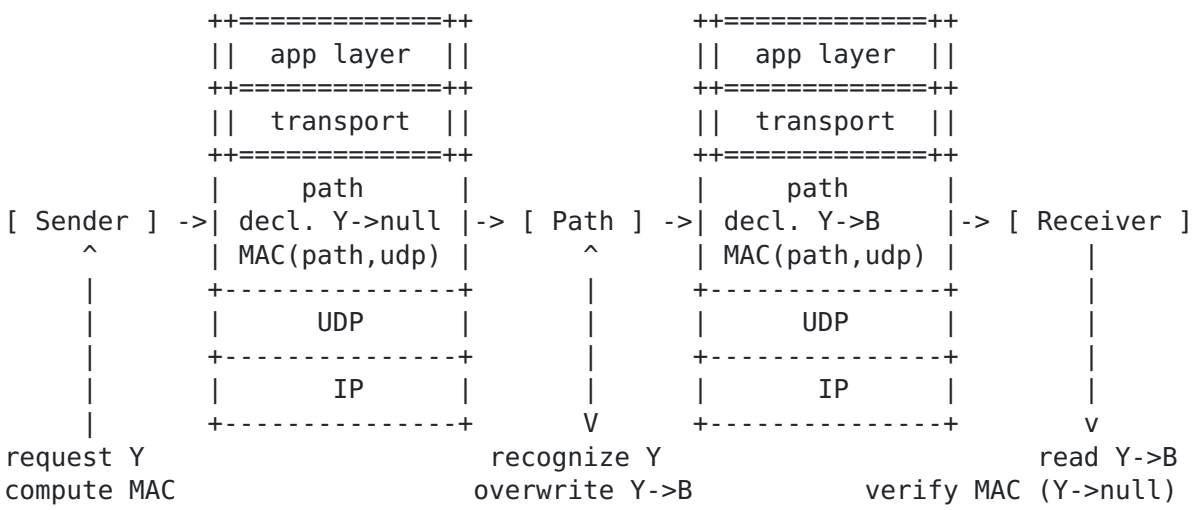

Figure 2: Path to Receiver Declaration

This mechanism allows the sender to allow the receiver to receive path declarations. However, if it is the sender that needs to know the final result of the path declaration, this can be fed back to the sender over an encrypted channel. Depending on the characteristics of the upper layer, this encrypted channel can either be provided by the upper layer, or be provided by the layer implementing the mechanisms, using a key derived from the same secret known only to the endpoints used to generate the MAC. The fact that a path-to-receiver declaration should be fed back to the sender is part of the definition of the key.

This arrangement is illustrated in Figure 3.

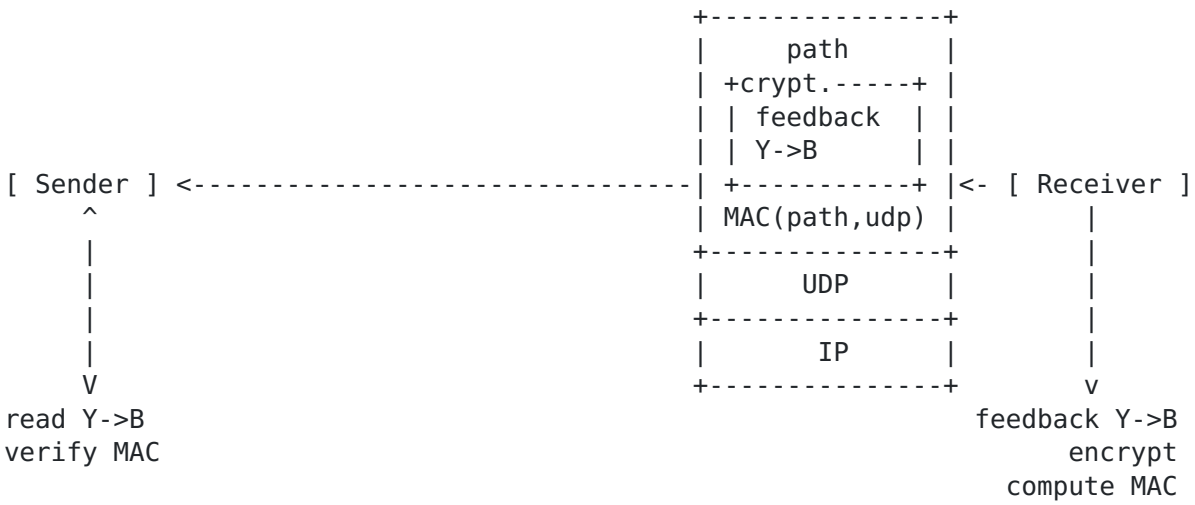

Figure 3: Receiver Feedback

2.3. Direct Path-to-Sender Declarations

Path-to-receiver declaration is impossible if a path element will drop a packet. In order to allow a path element to provide information about why a packet was dropped, it can send back a packet containing only a path-to-sender declaration. The fact that this is a direct path-to-sender declaration is part of the definition of the key. A path-to-sender declaration packet can only contain path-to-sender declarations. Since in the general case the path element has no shared secret with which to generate a MAC, this declaration cannot be integrity protected.

In order for a path-to-sender declaration to traverse any network address translation (NAT) function along the path, the path element must send the packet with the IP addresses and transport/ encapsulation layer ports reversed.

The sender must indicate it is willing to receive path-to-sender declarations, and this indication must include some nonce or other identifier that is hard to guess by devices not on path, which is returned with the path-to-sender declaration to identify the packet to which the declaration applies.

Only one path-to-sender declaration packet may be sent per dropped packet; this mitigates the abuse of this mechanism for executing amplified reflection attacks.

This arrangement is illustrated in Figure 4.

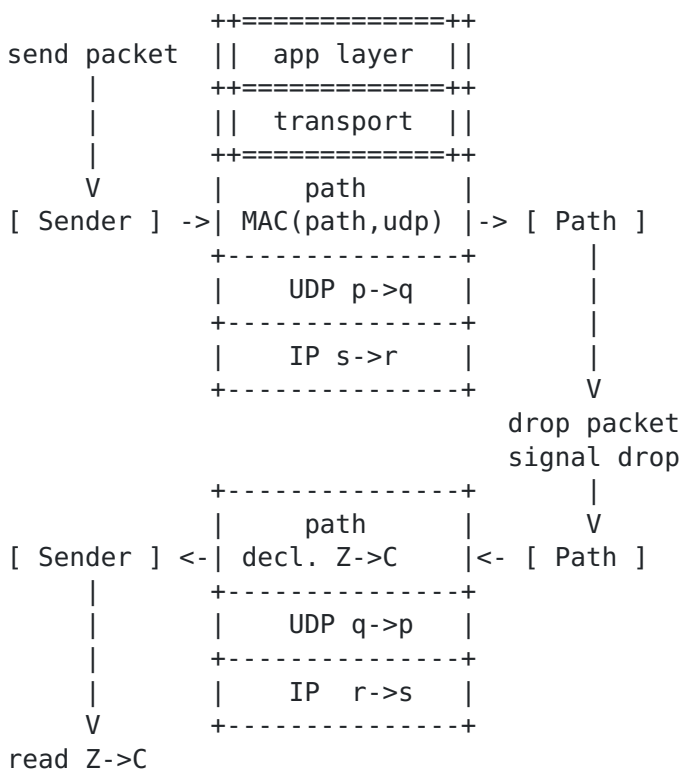

Figure 4: Direct Path to Sender Declarations

3. Technical Considerations

A few details must be considered in the implementation of the mechanisms described above; some are general, and some apply only in specific circumstances. They are described in the subsections below.

3.1. Cryptographic Context Bootstrapping

These mechanisms rely on an upper layer to establish a cryptographic context in order to establish a shared secret from which MAC keys can be derived. This cryptographic state may be established with each transport session or may be resumable across multiple transport sessions, depending on the upper layer's design. If no context exists, though, the integrity of the declarations made via these mechanisms cannot be protected by MAC. We propose two possible solutions to this situation:

1. The mechanisms can be implemented such that MAC is mandatory. In this arrangement, no sender-to-path and/or path-to-receiver declarations can be made until cryptographic context is

bootstrapped. The vocabulary of declarations can therefore not include declarations that must be sent on the first packet.

2. The mechanisms can be implemented such that the MAC is eventual. In this arrangement, sender-to-path declarations can be made before cryptographic context establishment, but are open to undetected modification along the path; path-to-receiver declarations are not allowed before cryptographic context establishment. A MAC for previous sender-to-path declarations must be sent after cryptographic context establishment; lack of receiving this MAC within a defined (and small) number of packets from the sender is treated by the receiver as verification failure and leads to transport association reset.

3.2. Adding Integrity and Confidentiality Protection Along the Path

If a path element and the sender share some cryptographic context through some out-of-band means, sender to path declarations can also be integrity protected using a MAC generated by the sender and carried within the declaration itself. In this case, if the path element fails to verify the MAC, it simply ignores the declaration.

Similarly, if a path element and the receiver share some cryptographic context through some out-of-band means, path to receiver declarations can also be integrity protected using a MAC generated by the path element and carried within the declaration itself. The use of a MAC is part of the definition of the key. In this case, if the receiver fails to verify the MAC, it causes a transport association reset.

This design pattern can also be used to address sender-to-path declarations to specific path elements: a declaration with an encrypted value is inherently addressed to only those path elements that possess the private or secret key to decrypt the value.

In each of these cases, the presence of a MAC within the declaration value and/or encryption of the declaration value is part of the definition of the key. Further definition of mechanisms for building cryptographic protocols over these mechanisms is out of scope for this document.

4. IANA Considerations

This document has no actions for IANA. A future document specifying a concrete implementation of these mechanisms and a vocabulary of declarations may create and modify an IANA registry of such declarations. [EDITOR'S NOTE: please remove this section at publication.]

5. Security Considerations

This document describes abstract mechanisms by which endpoints can share information about traffic flows with devices along the path, replacing the functions currently performed using traffic inspection of cleartext transport headers with explicit exposure when those headers are encrypted. We consider four potential threats against the design of these abstract mechanisms:

1. Injection of packets or declarations by an on-path attacker

2. Injection of packets declarations by an off-path attacker

3. Sender fingerprinting or other inference about the content of encrypted communications by an on-path attacker

5.1. Defending against On-Path Injection of Declarations

The MAC generated by a sending endpoint protects against on-path injection of declarations not authorized by the sender, or an on-path device spoofing the sending endpoint. Even if one on-path device manages to spoof a declaration to a device further along the path, MAC verification failure at the receiving endpoint will lead to upper layer association reset.

5.2. Defending against Off-Path Injection of Declarations

Since MAC verification requires the receiving endpoint, it may be possible for an off-path attacker to spoof a declaration that an on-path device would not be able to verify. In order to defend against this threat, the mechanisms should be implemented by exposing a hard-to-guess token selected by a sending endpoint and verified by a receiving endpoint as well as by on-path devices. This token may itself take the form of a declaration, or appear in a header enclosing the set of declarations.

5.3. Defending against fingerprinting attacks and overexposure

Since these abstract mechanisms are designed to explicitly expose metadata about encrypted traffic, the concern naturally arises that overexposure of metadata can be used to infer information about the type or content of encrypted information, or that information radiated off a sending endpoint can be used to create a fingerprint of that sender.

In order to defend against this threat, the vocabulary of signals to be used with this mechanism must be designed in a restrictive way:

- Definition of obviously-overexposing signals must be prohibited by the process used to define the vocabulary. Examples of obvious overexposure include sharing end-to-end secrets with on-path devices, tagging flows with a globally unique ID that can be associated with a sender (e.g. for mobility applications), or exposure of endpoint location at high resolution.

- Signals must be defined to use as few bits on the wire as possible, in order to reduce the cross section of the path layer packet header that can be used for fingerprinting, or potentially abused to coerce endpoints to add more information about their traffic.

The first restriction can be implemented using an IANA registry for the vocabulary of signals with a restrictive policy for addition of signals such as Standards Action [RFC5226], as well as a purposefully restricted codepoint space. The second restriction can also be assisted by defining a small maximum per-packet size for signals exposed using the mechanism, which would also have overhead benefits.

We note that by replacing present plain-text transport headers with encrypted transport headers, and allowing sending endpoints to explicitly expose (or not expose) information about those, that the cross section available for fingerprinting with these abstract mechanisms is much smaller than that presented by the current TCP/IP stack; see [I-D.trammell-privsec-defeating-tcpip-meta].

6. Acknowledgments

This work is 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.

Thanks to Aaron Falk, Ted Hardie, Joe Hildebrand, Mirja Kuehlewind, Natasha Rooney, Kyle Rose, and the participants at the PLUS BoF at IETF 96 in Berlin for the conversations leading to and informing the publication of this document. Special thanks to Mark Nottingham for posing the questions that led to the design space for the mechanisms described herein.

7. Informative References

[I-D.trammell-privsec-defeating-tcpip-meta]
Trammell, B., "Detecting and Defeating TCP/IP Hypercookie Attacks", draft-trammell-privsec-defeating-tcpip-meta-00 (work in progress), July 2016.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <http://www.rfc-editor.org/info/rfc4301>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <http://www.rfc-editor.org/info/rfc7049>.

Author's Address

Brian Trammell
ETH Zurich
Gloriastrasse 35
8092 Zurich
Switzerland

Email: ietf@trammell.ch