Network Working Group
Internet-Draft
Intended status: Informational
Expires: November 8, 2015

B. Trammell
ETH Zurich

May 07, 2015

Thoughts a New Transport Encapsulation Architecture draft-trammell-stackevo-newtea-01

Abstract

This document explores architectural considerations for using encapsulation in support of stack evolution and new transport protocol deployment in an increasingly encrypted Internet. These architectural considerations are based on an idealized architecture where all interactions among applications, endpoints, and the path occur explicitly, with this cooperation enforced cryptographically. This idealized architecture is then lensed through the state of devices in the present Internet and how they would impair the deployability of such an architecture, in order to support an incremental deployment of this approach.

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 November 8, 2015.

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

1. Introduction and Background

The current work of the IAB IP Stack Evolution Program is to support the evolution of the Internet's transport layer and its interfaces to other layers in the Internet Protocol stack. The need for this work is driven by two trends. First is the development and increased deployment of cryptography in Internet protocols to protect against pervasive monitoring [RFC7258], which will break many middleboxes used in the operation and management of Internet-connected networks and which assume access to plaintext content. An additional encapsulation layer to allow selective, explicit metadata exchange between the endpoints and devices on path to replace ad-hoc packet inspection is one approach to retain network manageability in an encrypted Internet.

Second is the increased deployment of new applications (e.g. interactive media as in RTCWEB [I-D.ietf-rtcweb-overview]) for which the abstractions provided by today's transport APIs (i.e., either a single reliable stream as in SOCK_STREAM over TCP, or an unreliable, unordered packet sequence as in SOCK_DGRAM over UDP) are inadequate. This evolution is constrained by the presence of middleboxes which interfere with connectivity or packet invariability in the presence of new transport protocols or transport protocol extensions.

Parts of this problem are presently being addressed in various ways by the IETF. The Transport Services (TAPS) Working Group is defining a new abstract interface for specifying transport requirements to the transport layer, with a vocabulary based upon existing transport protocol service features. This will allow future transport layers (implemented in userspace libraries, in operating system kernels, or some combination of the two) to select a wire protocol based upon these requirements and the properties of the path between the endpoints, including the impairments of middleboxes along that path.

The Substrate Protocol for User Datagrams (SPUD) Birds of a Feather (BoF) session at IETF 92 in Dallas in March 2015 discussed use cases and a prototype protocol [I-D.hildebrand-spud-prototype] for encapsulating opaque content in UDP, with a facility for signaling limited transport semantics and binding metadata to packets and flows in a flexible way. This encapsulation is designed to provide explicit cooperation between endpoints and middleboxes where this makes sense, while allowing new transport protocol development to

happen both in the kernel - to which it has largely been restricted due to the history of the development of TCP/IP - as well as in userspace. The outcome of the BoF session was to continue the discussion about the architecture, transport semantics and metadata vocabulary, and experimental implementation of this approach on the mailing list established for the BoF (spud@ietf.org)

SPUD is not the only protocol-level work to address explicit communication between endpoints and devices along the path: work in the Transport Layer Security (TLS) working group [I-D.huitema-tls-dtls-as-subtransport] discusses the possibility and provides a gap analysis for running a "minimal common subtransport" exposing common transport semantics as in SPUD directly over the Datagram Transport Layer Security (DTLS) protocol [RFC6347].

These efforts aim at building flexible mechanisms to solve the problem of expanding the interface between the transport layer and the applications above it as well as the problem of making explicit the contract between the transport layer and devices on path which should, in an end-to-end Internet, limit themselves to lower-layer interactions, but practically speaking have not done so for the past two decades.

This document aims to provide an architectural basis for these efforts, enumerating a set of architectural assumptions for transport evolution based upon new encapsulations, and discussing limitations on the vocabulary used in each of these new interfaces necessary to achieve deployment.

2. Terminology

This document borrows terminology from [I-D.ietf-taps-transports], specifically Transport Service, Transport Service Feature, Transport Protocol, and Transport Protocol Component, for discussing the composition of transport services.

[EDITOR'S NOTE: define Application Endpoint, Endhost, and Routable Endpoint, as well as Midpoint, Middlebox, etc., using existing terminology where applicable. A defined terminology here will help avoid imprecision in this conversation.]

3. An Architecture for Explicit Path-Endpoint Cooperation

The present Internet architecture is rife with implicit cooperation between endpoints and devices on the path between them. For example, network address translators (NATs) rewrite addresses and ports in order to increase the size of the Internet at the expense of universal reachability, but this translation is not explicitly

invoked by either endpoint. Traffic classification is often required for network management purposes, and often uses deep packet inspection to determine the traffic class of a particular packet or flow.

It is this implicit cooperation which has led to the ossification of the transport layer in the Internet. Implicit cooperation requires devices along the path to make assumptions about the format of the packets and the nature of the traffic they are forwarding, which in turn leads to problems using protocols which don't meet these assumptions. It also forces application and transport protocol developers to build protocols that operate in this presumed, least-common-denominator network environment.

We take the position that this situation can be improved by replacing implicit cooperation with explicit cooperation. We first explore the properties of an ideal architecture for explicit cooperation, then consider the constraints imposed by the present configuration of the Internet which would make transition to this ideal architecture infeasible. From this we derive a set of architectural principles and protocol design requirements which will support an incrementally deployable approach to explicit cooperation between applications on endpoints and middleboxes in the Internet.

3.1. Principles: What does good look like?

We can take some guidance for the future from the original Internet architecture.

The original Internet architecture defined the split between TCP and IP by defining IP to contain those functions the gateways need to handle (and possibly de- and re-encapsulate, including fragmentation), while defining TCP to contain functions that can be handled by hosts end-to-end [RFC0791]. Gateways were essentially trusted not to meddle in TCP.

As a first principle, a strict division between hop-to-hop and end-to-end functions is desirable to restore and maintain end-to-end service in the Internet.

In the original architecture, there was no provision for "in-network functionality" beyond forwarding, fragmentation, and basic diagnostics. This was as much a function of adherence to the end-to-end-principle [Saltzer84] as a desire to limit computational complexity and state requirements on the gateways.

In-network functions in the Internet Protocol as presently defined are explicit. Forwarding is inherently explicit: placing an address

in the destination address field, the endpoint (and by extension, the application) indicates that a packet should be sent to a given address. The contract for fragmentation was implicit in IPv4, but in-network fragmentation was removed in IPv6 [RFC2460].

We note that layer boundaries can be enforced using sufficiently strong cryptography.

As a second principle, the presence of in-network functionality along a path which results in the modification of packet streams received at the other end of a connection should be explicit.

For optional functionality, the applications at either end of a connection should have appropriate explicit control over the presence of those functions on path, even if they are present by default. For those functions which are necessary, without which there is no end-to-end connectivity (e.g. NATs in many environments), or which are otherwise non-optional for operational reasons (e.g. firewalls), the functionality should be explicitly discoverable by the applications on either end.

This explicitness extends into the transport stack in the endpoint. When applications can clearly define transport requirements, instead of implicitly lensing them through known implementations of each socket type, these transport requirements can be exposed to and/or matched with properties of devices along the path, where that is useful.

[EDITOR'S NOTE: this is perhaps a bit further than we want to actually go, but this would seem to be the logical conclusion of "make path interaction explicit"]

3.2. Impairments: What keeps us from getting there?

The clear separation of network and transport layer has been steadily eroded over the past twenty years or so. Network address and port translation (NAPT) have effectively made the first four bytes of the transport header a de-facto part of the network layer, and have made it difficult to deploy protocols where NAPT devices don't know that the ports are safe to touch: anything other than UDP and TCP. Protocols to support with NAT traversal (e.g. Interactive Connectivity Establishment [RFC5245]) do not address this fundamental problem.

Mechanisms that could be used to support explicit cooperation between applications and middleboxes could be supported within the network layer. The IPv6 Hop-by-Hop Options Header is explicitly intended for this purpose, and a new hop-by-hop option could be defined. However,

there are some limitations to using this header: it is only supported by IPv6, it may itself cause packets to be dropped, it may not be handled efficiently (or indeed at all) by currently deployed routers and middleboxes, and it requires changes to operating system stacks at the endpoints to allow applications to access these headers.

One of the effects of the fact that cryptography enforces layer boundaries is that applications and transports run over HTTPS de facto [I-D.blanchet-iab-internetoverport443], since HTTPS is the most widely implemented, accessible, and deployable way for application developers to get this enforcement.

However, the greatest barriers to explicit cooperation between applications and devices along the path is the lack of explicit trust among them. While it is possible to assign trust within the "first hop" administrative domains, especially when the endpoint and network operator are the same entity, building and operating an infrastructure for assigning and maintaining these trust relationships within an Internet context is currently impractical.

Finally, the erosion of the end-to-end principle has not occurred in a vacuum. There are incentives to deploy in-network functions, and services that are impaired by them have already worked around these impairments. For example, the present trend toward service recentralization ("cloud computing") can be seen in part as the market's response to the end of end-to-end. Tf every application-layer transaction is mediated by services owned by the application's operator, two-end NAT traversal is no longer important. This new architecture for services has additional implications for the types of interactions supported, and for the types of business models encouraged, which may in turn make some of the concerns about limited deployability of new transport protocols moot.

3.3. What can we do?

First we turn to the problem of re-separation of the network layer from the transport layer. NAPT, as noted, has effectively made the ports part of the network layer, and this change is not easy to undo, so we can make this explicit. In many NAPT environments only UDP and TCP traffic will be forewarded, and a packet with a TCP header may be assumed by middleboxes to have TCP semantics; therefore, the solution space is constrained to putting the "new" separation between the network and transport layers within a UDP encapsulation. This has a further positive implication for incremental deployability: it is possible to implement UDP-based encapsulations in userspace

4. Encapsulation and signaling mechanisms

[EDITOR'S NOTE: frontmatter on encaps]

4.1. Encapsulations are narrow

A good deal of experience with tunnels has shown that the per-stream overhead of a given encapsulation is generally less important than its impact on MTU. For instance, the SPUD prototype as presently defined needs at least 20 additional bytes in the header per packet: 2 each for source and destination UDP port, 2 for UDP length, 2 for UDP checksum, 8 to identify tubes, 1 for control bits for SPUD itself, and 3 for the smallest possible CBOR map containing a single opaque higher layer datagram. For 1500-byte Ethernet frames, the marginal cost of SPUD before is therefore 1.33% in byte terms, but it does imply that 1450 byte application datagrams will no longer fit in a single SPUD-over-UDP-over-IPv4-over Ethernet packet.

This fact has two implications for encapsulation design: First, maximum payload size per packet should be communicated up to the higher layer, as an explicit feature of the transport layer's interface. Second, link-layer MTU is a fundamental property of each link along a path, so any signaling protocol allowing path elements to communicate to the endpoint should treat MTU as one of the most important properties along the path to explicitly signal.

5. Implicit trust in endpoint-path signaling

In a perfect world, the trust relationships among endpoints and elements on path would be precisely and explicitly defined: an endpoint would explicitly delegate some processing to a network element on its behalf, network elements would be able to trust any command from any endpoint, and the integrity and authenticity of signaling in both directions would be cryptographically protected.

However, both the economic reality that the users at the endpoints and the operators of the network may not always have aligned interests, as well as the difficulty of universal key exchange and trust distribution among widely heterogeneous devices across administrative domain boundaries, require us to take a different approach toward building deployable, useful metadata signaling.

We observe that imperative signaling approaches in the past have often failed in that they give each actor incentives to lie. Markings to ask the network to explicitly treat some packets as more important than others will see their value inflate away - i.e., most packets from most sources will be so marked - unless some mechanism is built to police those markings. Reservation protocols suffer from

the same problem: for example, if an endpoint really needs 1Mbps, but there is no penalty for reserving 1.5Mbps "just in case", a conservative strategy on the part of the endpoint leads to over-reservation.

5.1. Declarative marking

An alternate approach is to treat these markings as declarative and advisory, and to treat all markings on packets and flows as relative to other markings on packets and flows from the same sender. In this case, where neither endpoints nor path elements can reliably predict the actions other elements in the network will take with respect to the marking, and where no endpoint can ask for special treatment at the expense of other endpoints, the incentives to marking inflation are greatly diminished.

5.2. Verifiable marking

Second, markings and declarations should be defined in such a way that they are verifiable, and verification built end to endpoints and middleboxes wherever practical. Suppose for example an endpoint declares that it will send constant-bitrate, loss-insensitive traffic at 192kbps. The average data rate for the given flow is trivially verifiable at any endpoint. A firewall which uses this data for traffic classification and differential quality of service may spot-check the data rate for such flows, and penalize those flows and sources which are clearly mismarking their traffic.

It is probably not possible, especially in an environment with ubiquitous opportunistic encryption [RFC7435], to define a useful marking vocabulary such that every marking will be so easily verifiable. However, in an environment in which markings are variably-trusted and verified, trustworthiness can be dynamically assigned by each device, as well as

the trustworthiness of each endpoint and path can be independently assessed by any node involved in a communication, and reputation-tracking approaches can be used to signal how believable a declaration is to transport protocols which use those declarations, as well as to provide an additional incentive to mark honestly.

6. IANA Considerations

This document has no considerations for IANA.

7. Security Considerations

This revision of this document presents no security considerations. A more rigorous definition of the limits of declarative and verifiable marking would need to be evaluated against a specified threat model, but we leave this to future work.

8. Acknowledgments

Many thanks to the attendees of the IAB Workshop on Stack Evolution in a Middlebox Internet (SEMI) in Zurich, 26-27 January 2015; most of the thoughts in this document follow directly from discussions at SEMI. This work is partially supported by the European Commission under Grant Agrement FP7-318627 mPlane; support does not imply endorsement by the Commission of the content of this work.

9. Informative References

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, May 2014.

[RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", RFC 7435, December 2014.

[I-D.ietf-rtcweb-overview] Alvestrand, H., "Overview: Real Time Protocols for Browser-based Applications", draft-ietf-rtcweb-overview-13 (work in progress), November 2014.

[I-D.ietf-taps-transports] Fairhurst, G., Trammell, B., and M. Kuehlewind, "Services provided by IETF transport protocols and congestion control mechanisms", draft-ietf-taps-transports-03 (work in progress), February 2015.

[I-D.hildebrand-spud-prototype]
Hildebrand, J. and B. Trammell, "Substrate Protocol for User Datagrams (SPUD) Prototype", draft-hildebrand-spud-prototype-03 (work in progress), March 2015.

[I-D.huitema-tls-dtls-as-subtransport]
Huitema, C., Rescorla, E., and J. Jana, "DTLS as Subtransport protocol", draft-huitema-tls-dtls-as-subtransport-00 (work in progress), March 2015.

[I-D.blanchet-iab-internetoverport443]
Blanchet, M., "Implications of Blocking Outgoing Ports Except Ports 80 and 443", draft-blanchet-iab-internetoverport443-02 (work in progress), July 2013.

[Saltzer84]
Saltzer, J., Reed, D., and D. Clark, "End-to-End Arguments in System Design (ACM Trans. Comp. Sys.)", 1984.

Author's Address

Brian Trammell
ETH Zurich
Gloriastrasse 35
8092 Zurich
Switzerland

Email: ietf@trammell.ch