Network Working Group
Internet-Draft
Intended status: Informational
Expires: March 26, 2016

B. Trammell, Ed.
ETH Zurich

September 23, 2015

Architectural Considerations for Transport Evolution with Explicit Path Cooperation

draft-trammell-stackevo-explicit-coop-00

Abstract

The IAB Stack Evolution in a Middlebox Internet (SEMI) workshop in Zurich in January 2015 and the follow-up Substrate Protocol for User Datagrams (SPUD) BoF session at the IETF 92 meeting in Dallas in March 2015 identified the potential need for a UDP-based encapsulation to allow explicit cooperation with middleboxes while using encryption at the transport layer and above to protect user payload and metadata from inspection and interference. This document proposes a set of architectural considerations for such approaches.

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 March 26, 2016.

1. Introduction and Motivation

The IAB IP Stack Evolution Program aims 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 protocols (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.

The core issue is one of layer violation. The boundary between the network and transport layers was originally defined to be the boundary between information used (and potentially modified) hop-by-hop, and that only used end-to-end. The widespread deployment of network address and port translation (NAPT) in the Internet has eroded this boundary. The first four bytes after the IP header or header chain - the source and destination ports - are now the de facto boundary between the layers. This erosion has continued into the transport and application layer headers and down into content, as the capabilities of deployed middleboxes have increased over time. Evolution above the network layer 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, so we need a new approach.

This boundary can be reinforced by encapsulating new transport protocols in UDP and encrypting everything above the UDP header.

However, this brings with it other problems. First, middleboxes which maintain state must use timers to expire that state for UDP flows, since there is no exposure of flow lifetime and bidirectional establishment as with TCP's SYN, ACK, FIN, and RST flags. These timers are often set fast enough to require a relatively high rate of heartbeat traffic to maintain this state. A limited facility to expose basic semantics of the underlying transport protocol would allow these devices to keep state as they do with TCP, with no worse characteristics with respect to state management than those of TCP.

This is a specific case of a more general issue: some of the inspection of traffic done by middleboxes above the network-transport boundary is operationally useful. However, the use of transport layer and higher layer headers is an implicit feature of the architecture: middleboxes are exploiting the fact that these headers are transmitted in cleartext. There is no explicit cooperation here: the endpoints have no control over the information exposed to the path, and the middleboxes no information about the intentions of the endpoint application other than that inferred from the inspected traffic. We propose a change to the architecture to remedy this condition.

2. Explicit Cooperation as Architectural Principle

The principle behind this change is one of explicit cooperation.

The present Internet architecture is rife with implicit cooperation between endpoints and devices on the path between them. 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.

This situation can be improved by making this cooperation explicit. 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.

2.1. 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. 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. Fragmentation was implicit in IPv4, though in-network fragmentation was removed in IPv6 [RFC2460]. This was as much a function of adherence to the end-to-end-principle [Saltzer84] as it was an engineering reaction to limited computational and state capacity on the gateways.

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

As a second principle, in-network functionality along a path which results in the modification of packet streams received at the other end of a connection should be explicitly visible (and, where appropriate to the nature of the functionality, controllable) by the endpoints.

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.

2.2. 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 NAT traversal (e.g. Interactive Connectivity Establishment [RFC5245]) do not address this fundamental issue.

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 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 [I-D.baker-6man-hbh-header-handling], 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 can be seen in part as the market's response to the end of end-to-end. If 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.

2.3. 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 forwarded, and a packet with a TCP header may be assumed by middleboxes to have TCP semantics; therefore, the solution space is most probably constrained to putting the "new" separation between the network and transport layers within a UDP encapsulation.

Since the relative delay in getting new transport protocols into widely deployed kernel implementations has historically been another impediment to deploying these new protocols in the Internet, a UDP encapsulation based approach has a further implication for incremental deployability: it is possible to implement UDP-based encapsulations in userspace. While userspace implementations may lack some of the interfaces to lower layers available to kernelspace implementations necessary to build high-performance transport implementations, the ability to deploy widely and quickly may help break the chicken-and-egg problem of getting a transport protocol adopted into kernelspace.

To support explicit cooperation in an environment where trust relationships are variable and there may be no context with which to authenticate every device along a path with which a

3. Properties of encapsulation and signaling mechanisms

We now turn to observations about and probable constraints on an encapsulation and signaling-based approaches to explicit cooperation. These thoughts are presently unordered, some having come from initial efforts at defining requirements for SPUD.

3.1. The narrowness of encapsulation

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.

3.2. 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.

3.3. 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.

3.4. 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.

We note that it is optimistic to assume, especially in an environment with ubiquitous opportunistic encryption [RFC7435], that it is possible to define a useful marking vocabulary such that every marking will be so easily verifiable. However, using verifiability as a design goal can lead to marking vocabularies which are less likely, on balance, to be be gameable and gamed.

3.5. Privacy Tradeoffs in Metadata Signaling

Protocol engineering experience has shown that extensibility is a key design goal for new protocols: especially in this case, an attempt to "de-ossify" the protocol stack is really an attempt to "re-ossify" it around new realities and requirements, so it makes sense to ensure this effort must not be repeated. However, extensibility brings with it the potential for adding new metadata which can be used to increase linkability and surveillability of traffic. Identifiers used internally by the signaling mechanism may also increase linkability. Care must be taken when designing identifier spaces and extensibility mechanisms to minimize these risks.

3.6. In-band, out-of-band, piggybacked, and interleaved signaling

Signaling channels may be in-band (that is, using the same 5 tuple as the encapsulated traffic) or out-of-band (using another channel and different flows). Here there are also many tradeoffs to consider. In-band signaling has the advantage that it does not require foreknowledge of the identity and addresses of devices along the path by endpoints and vice versa, but does add complexity to the signaling protocol.

In-band signaling can be either piggybacked on the overlying transport or interleaved with it. Piggybacked signaling uses some number of bits in each packet generated by the overlying transport to achieve signaling. It requires either reducing the MTU available to the encapsulated transport and/or opportunistically using bits

between the network-layer MTU and the bits actually used by the transport.

This is even more complicated in the case of middleboxes that wish to add information to piggybacked-signaling packets, and may require the endpoints to introduce "scratch space" in the packets for potential middlebox signaling use, further increasing complexity and overhead.

In contrast, interleaved signaling uses signaling packets on the same 5-tuple. This reduces complexity and sidesteps MTU problems, but is only applicable when the signaling can be considered valid for the entire flow or bound to some subset of packets in the flow via an identifier.

Out-of-band signaling uses direct connections between endpoints and middleboxes, separate from the encapsulated transport - connections that are perhaps negotiated by in-band signaling. A key disadvantage here is that out-of-band signaling packets may not take the same path as the packets in the encapsulated transport; therefore connectivity cannot be guaranteed.

Signaling of path-to-endpoint information, in the case that a middlebox wants to signal something to the sender of the packet, raises the added problem of either (1) requiring the middlebox to send the information to the receiver for later reflection back to the sender, which has the disadvantage of complexity, or (2) requiring out-of-band direct signaling back to the sender, which in turn either requires the middlebox to spoof the source address and port of the receiver to ensure equivalent NAT treatment, or some other NAT-traversal approach.

The tradeoffs here must be carefully weighed, and a comprehensive approach may use a mix of all these communication patterns.

3.7. Reflection protection and return routability

The ease of forging source addresses in UDP together with the only limited deployment of network egress filtering [RFC2827] means that UDP traffic presently lacks a return routability guarantee. This leads to UDP being useful for reflection and amplification attacks, which in turn has led in part to the present situation wherein UDP traffic may be blocked by firewalls when not explicitly needed by an organization as part of its Internet connectivity.

Return routability is therefore a minimal property of any transport that can be responsibly deployed at scale in the Internet.

4. IANA Considerations

This document has no actions for IANA.

5. 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.

6. 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 and the subsequent SPUD BoF in Dallas in March 2015. Some text for this revision of this document has been taken from [I-D.trammell-spud-req], written with Mirja Kuehlewind, David Black, Ken Calvert, Ted Hardie, Joe Hildebrand, Jana Iyengar, and Eric Rescorla. 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.

7. 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, DOI 10.17487/RFC5245, April 2010, <http://www.rfc-editor.org/info/rfc5245>.

[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, DOI 10.17487/RFC7258, May 2014, <http://www.rfc-editor.org/info/rfc7258>.

[RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", RFC 7435, DOI 10.17487/RFC7435, December 2014, <http://www.rfc-editor.org/info/rfc7435>.

[I-D.ietf-rtcweb-overview]
Alvestrand, H., "Overview: Real Time Protocols for Browser-based Applications", draft-ietf-rtcweb-overview-14 (work in progress), June 2015.

[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-06 (work in progress), July 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.

[I-D.baker-6man-hbh-header-handling]
Baker, F., "IPv6 Hop-by-Hop Header Handling", draft-baker-6man-hbh-header-handling-02 (work in progress), July 2015.

[I-D.trammell-spud-req]
Trammell, B. and M. Kuehlewind, "Requirements for the design of a Substrate Protocol for User Datagrams (SPUD)", draft-trammell-spud-req-00 (work in progress), July 2015.

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

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, DOI 10.17487/RFC0792, September 1981, <http://www.rfc-editor.org/info/rfc792>.

[RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, <http://www.rfc-editor.org/info/rfc2827>.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, March 2007.

[RFC7510] Xu, X., Sheth, N., Yong, L., Callon, R., and D. Black, "Encapsulating MPLS in UDP", RFC 7510, DOI 10.17487/RFC7510, April 2015, <http://www.rfc-editor.org/info/rfc7510>.

[I-D.trammell-stackevo-newtea] Trammell, B., "Thoughts a New Transport Encapsulation Architecture", draft-trammell-stackevo-newtea-01 (work in progress), May 2015.

[I-D.iab-semi-report] Trammell, B. and M. Kuehlewind, "IAB Workshop on Stack Evolution in a Middlebox Internet (SEMI) Report", draft-iab-semi-report-01 (work in progress), July 2015.

[I-D.ietf-dart-dscp-rtp] Black, D. and P. Jones, "Differentiated Services (DiffServ) and Real-time Communication", draft-ietf-dart-dscp-rtp-10 (work in progress), November 2014.

Author's Address

Brian Trammell (editor)
ETH Zurich
Gloriastrasse 35
8092 Zurich
Switzerland

Email: ietf@trammell.ch