Network Working Group B. Trammell, Ed.
Internet-Draft M. Kuehlewind, Ed.
Intended status: Informational ETH Zurich
Expires: January 7, 2016 July 06, 2015

Requirements for the design of a Substrate Protocol for User Datagrams (SPUD)
draft-trammell-spud-req-00

Abstract

The 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 protocol to allow explicit cooperation with middleboxes while using new, encrypted transport protocols. This document proposes an initial set of requirements for such a protocol, and discusses tradeoffs to be made in further refining these requirements.

1. Motivation

A number of efforts to create new transport protocols or experiment with new network behaviors have been built on top of UDP, as it traverses firewalls and other middleboxes more readily than new protocols do. Each such effort must, however, either manage its flows within common middlebox assumptions for UDP or train the middleboxes on the new protocol (thus losing the benefit of using UDP). A common Substrate Protocol for User Datagrams (SPUD) would allow each effort to re-use a set of shared methods for notifying middleboxes of the flows' semantics, thus avoiding both the limitations of current flow semantics and the need to re-invent the mechanism for notifying the middlebox of the new semantics.

As a concrete example, it is common for some middleboxes to tear down required state (such as NAT bindings) very rapidly for UDP flows. By notifying the path that a particular transport using UDP maintains session state and explicitly signals session start and stop using the substrate, the using protocol may reduce or avoid the need for heartbeat traffic.

This document defines a specific set of requirements for a SPUD facility, based on analysis on a target set of applications to be developed on SPUD developing experience with a prototype described in [I-D.hildebrand-spud-prototype]. It is intended as the basis for determining the next steps to make progress in this space, including eventually chartering an working group for specific protocol engineering work.

2. History

An outcome of the IAB workshop on Stack Evolution in a Middlebox Internet (SEMI) [I-D.iab-semi-report], held in Zurich in January 2015, was a discussion on the creation of a substrate protocol to support the deployment of new transport protocols in the Internet. Assuming that a way forward for transport evolution in user space would involve encapsulation in UDP datagrams, the workshop noted that it may be useful to have a facility built atop UDP to provide minimal signaling of the semantics of a flow that would otherwise be available in TCP. At the very least, indications of first and last packets in a flow may assist firewalls and NATs in policy decision and state maintenance. Further transport semantics would be used by

the protocol running atop this facility, but would only be visible to
the endpoints, as the transport protocol headers themselves would be

encrypted, along with the payload, to prevent inspection or modification. This encryption might be accomplished by using DTLS [RFC6347] as a subtransport [I-D.huitema-tls-dtls-as-subtransport] or by other suitable methods. This facility could also provide minimal application-to-path and path-to-application signaling, though there was less agreement about what should or could be signaled here.

The Substrate Protocol for User Datagrams (SPUD) BoF was held at IETF 92 in Dallas in March 2015 to develop this concept further. It is clear from discussion before and during the SPUD BoF that any selective exposure of traffic metadata outside a relatively restricted trust domain must be advisory, non-negotiated, and declarative rather than imperative. This conclusion matches experience with previous endpoint-to-middle and middle-to-endpoint signaling approaches. As with other metadata systems, exposure of specific elements must be carefully assessed for privacy risks and the total of exposed elements must be so assessed. Each exposed parameter should also be independently verifiable, so that each entity can assign its own trust to other entities. Basic transport over the substrate must continue working even if signaling is ignored or stripped, to support incremental deployment. These restrictions on vocabulary are discussed further in [I-D.trammell-stackevo-newtea]. This discussion includes privacy and trust concerns as well as the need for strong incentives for middlebox cooperation based on the information that are exposed.

3. Terminology

This document uses the following terms

- Overlying transport: : A transport protocol that uses SPUD for middlebox signaling and traversal.

- Endpoint: : A node that sends or receives a packet. In this document, this term may refer to either the SPUD implementation on this node, the overlying transport implementation on this node, or the applications running over that overlying transport.

- Path: : The set of Internet Protocol nodes and links that a given packet traverses from endpoint to endpoint.

- Middlebox: : A device on the path that makes decisions about forwarding behavior based on other than IP or sub-IP addressing information, and/or that modifies the packet before forwarding.

4. Use Cases

The primary use case for endpoint to path signaling, making use of packet grouping, is the binding of limited related semantics (start-tube and stop-tube) to a group of packets which are semantically related in terms of the application or overlying transport. By explicitly signaling start and stop semantics, a flow allows middleboxes to use those signals for setting up and tearing down their relevant state (NAT bindings, firewall pinholes), rather than requiring the middlebox to infer this state from continued traffic. At best, this would allow the application to refrain from sending heartbeat traffic, which might result in reduced radio utilization (and thus greater battery life) on mobile platforms.

SPUD may also provide some facility for SPUD-aware nodes on the path to signal some property of the path relative to a tube to the endpoints and other SPUD-aware nodes on the path. The primary use case for path to application signaling is parallel to the use of ICMP [RFC0792], in that it describes a set of conditions (including errors) that applies to the datagrams as they traverse the path. This usage is, however, not a pure replacement for ICMP but a "5-tuple ICMP" which would traverse NATs in the same way as the traffic related to it, and be deliverable to the application with appropriate tube information.

[EDITOR'S NOTE: specific applications we think need this go here? reference draft-kuehlewind-spud-use-cases.]

5. Functional Requirements

The following requirements detail the services that SPUD must provide to overlying transports, endpoints, and middleboxes using SPUD.

5.1. Grouping of Packets

Transport semantics and many properties of communication that endpoints may want to expose to middleboxes are bound to flows or groups of flows (five-tuples). SPUD must therefore provide a basic facility for associating packets together (into what we call a "tube" ,for lack of a better term) and associate information to these groups of packets. The simplest mechanisms for association would involve the addition of an identifier to each packet in a tube. The tube ID must be bi-directional to support state establishment and is scoped to the forward and backward five-tuple due to privacy concern. Current thoughts on the tradeoffs on requirements and constraints on this identifier space are given in Section 7.1.

5.2. Endpoint to Path Signaling

SPUD must be able to provide information from the end-point(s) to all SPUD-aware nodes on the path. To be able to potentially communicate with all SPUD-aware middleboxes on the path SPUD must either be designed as an in-band signaling protocol, or there must be a pre-existing relationship between the endpoint and the SPUD-aware middleboxes along the path. Since it is implausible that an endpoint has these relationships to all SPUD-aware middleboxes on a certain path in the context of the Internet, SPUD must provide in-band signaling. SPUD may in addition also offer mechanisms for out-of-band signaling when it is appropriate to use. See Section 7.5 for more discussion.

5.3. Path to Endpoint Signaling

SPUD must be able to provide information from a SPUD-aware middlebox to the endpoint. See Section 7.5 for more discussion on tradeoffs here.

5.4. Extensibility

SPUD must enable multiple new transport semantics and application/path declarations without requiring updates to SPUD implementations in middleboxes.

5.5. Authentication

The basic SPUD protocol must not require any authentication or a priori trust relationship between endpoints and middleboxes to function. However, SPUD should support the presentation/exchange of authentication information in environments where a trust relationship already exists, or can be easily established, either in-band or out-of-band.

5.6. Integrity

SPUD must provide integrity protection of SPUD-encapsulated packets, though the details of this integrity protection are still open; see Section 7.3. At the very least, endpoints should be able to:

1. detect simple transmission errors over at least whatever headers SPUD uses its own signaling.

2. detect packet modifications by non-SPUD-aware middleboxes along the path

3. detect the injection of packets into a SPUD flow (defined by 5-tuple) or tube by nodes other than the remote endpoint.

The decision of how to handle integrity check failures other than case (1) may be left up to the overlying transport.

5.7. Privacy

SPUD must allow endpoints to control the amount of information exposed to middleboxes, with the default being the minimum necessary for correct functioning.

6. Non-Functional Requirements

The following requirements detail the constraints on how the SPUD facility must meet its functional requirements.

6.1. Middlebox Traversal

SPUD must be able to traverse middleboxes that are not SPUD-aware. Therefore SPUD must be encapsulated in a transport protocol that is known to be accepted on a large fraction of paths in the Internet, or implement some form of probing to determine in advance which transport protocols will be accepted on a certain path.

6.2. Low Overhead in Network Processing

SPUD must be low-overhead, specifically requiring very little effort to recognize that a packet is a SPUD packet and to determine the tube it is associated with.

6.3. Implementability in User-Space

To enable fast deployment SPUD and transports above SPUD must be implementable without requiring kernel replacements or modules on the endpoints, and without having special privilege (root or "jailbreak") on the endpoints. Usually all operating systems will allow a user to open a UDP socket. Therefore SPUD must provide an UDP-based encapsulation, either exclusively or as a mandatory-to-implement feature.

6.4. Incremental Deployability in an Untrusted, Unreliable Environment

SPUD must operate in the present Internet. In order to maximize deployment, it should also be useful as an encapsulation between endpoints even before the deployment of middleboxes that understand it. The information exposed over SPUD must provide incentives for

adoption by both endpoints and middleboxes, and must maximize privacy

(by minimizing information exposed). Further, SPUD must be robust to packet loss, duplication and reordering by the underlying network service. SPUD must work in multipath, multicast, and endpoint multi-homing environments.

Incremental deployability likely requires limitations of the vocabulary used in signaling, to ensure that each actor in a nontrusted environment has incentives to participate in the signaling protocol honestly; see [I-D.trammell-stackevo-newtea] for more.

6.5. Minimum restriction on the overlying transport

SPUD must impose minimal restrictions on the transport protocols it encapsulates. However, to serve as a substrate, it is necessary to factor out the information that middleboxes commonly rely on and endpoints are commonly willing to expose. This information should be included in SPUD, and might itself impose additional restrictions to the overlying transport. One example is that SPUD is likely to impose at least return routability and the presence of a feedback channel between endpoints, this being necessary for protection against reflection, amplification, and trivial state exhaustion attacks; see Section 7.4 for more.

6.6. Minimum Header Overhead

To avoid reducing network performance, the information and coding used in SPUD should be designed to use the minimum necessary amount of additional space in encapsulation headers.

6.7. No additional start-up latency

SPUD should not introduce additional start-up latency for overlying transports.

6.8. Reliability and Duplication

As any information provided by SPUD is anyway opportunistic, we assume for now that SPUD does not have to provide reliability and any SPUD mechanism using SPUD information must handle duplication of information. However, this decision also depends on the signal type used by SPUD, as further discussed in Section 7.5, and currently assumes that there are no SPUD information that would need to be split over multiple packets.

7. Open questions and discussion

The preceding requirements reflect the present best understanding of the authors of the functional and technical requirements on an encapsulation-based protocol for common middlebox-endpoint cooperation for overlying transports. There remain a few large open questions and points for discussion, detailed in the subsections below.

7.1. Tradeoffs in tube identifiers

Grouping packets into tubes requires some sort of notional tube identifier; for purposes of this discussion we will assume this identifier to be a simple vector of N bits. The properties of the tube identifier are subject to tradeoffs on the requirements for privacy, security, ease of implementation, and header overhead efficiency.

We first assume that the 5-tuple of source and destination IP address, UDP (or other transport protocol) port, and IP protocol identifier (17 for UDP) is used in the Internet as an existing flow identifier, due to the widespread deployment of network address and port translation. The question then arises whether tube identifiers should be scoped to 5-tuples (i.e., a tube is identified by a 6-tuple including the tube identifier) or should be separate, and presumed to be globally unique.

If globally unique, N must be sufficiently large to reduce to negligibility the probability of collision among multiple tubes having the same identifier along the same path during some period of time. An advantage of globally unique tube identifiers is they allow migration of per-tube state across multiple five-tuples for mobility support in multipath protocols. However, globally unique tube identifiers would also introduce new possibilities for user and node tracking, with a serious negative impact on privacy.

In the case of 5-tuple-scoped identifiers, mobility must be supported separately by each overlying transport. N must still be sufficiently large, and the bits in the identifier sufficiently random, that possession of a valid tube ID implies that a node can observe packets belonging to the tube. This reduces the chances of success of blind packet injection attacks of packets with guessed valid tube IDs.

Further using multiple tube identifiers within one 5-tuple also raises some protocol design questions: Can one packet belong to multiple tubes? Do all packets in a five-tuple flow have to belong

to one tube? Can/Must the backward flow have the same tube ID or a different one? Especially at connection start-up, depending on the

semantics of the overlying transport protocol, there likely might be only one packet to start multiple streams/tubes. Can this start message signal multiple tube IDs at once or do we need an own start message for each tube? Or is it in this case not possible to have multiple tubes within one five-tuple? These questions have to be further investigated based on the semantic of existing transport protocols. The discussion in [I-D.ietf-dart-dscp-rtp] concerning use of different QoS treatments within a single 5-tuple is related, e.g., WebRTC multiplexing of multiple application layer flows onto a single transport layer (5-tuple) flow.

7.2. Property binding

Related to identifier scope is the scope of properties bound to SPUD packets by endpoints. SPUD may support both per-tube properties as well as per-packet properties. Properties signaled per packet reduce state requirements at middleboxes, but also increase per-packet overhead. It is likely that both types of property binding are necessary, but the selection of which properties to bind how must be undertaken carefully.

7.3. Tradeoffs in integrity protection

In order to protect the integrity of information carried by SPUD against trivial forging by malicious devices along the path, it is necessary to be able to authenticate the originator of that information. We presume that the authentication of endpoints is a generally desirable property, and to be handled by the overlying transport; in this case, SPUD can borrow that authentication to protect the integrity of endpoint-originated information.

However, in the Internet, it is not in the general case possible for the endpoint to authenticate every middlebox that might see packets it sends and receives. In this case information produced by middleboxes may enjoy less integrity protection than that produced by endpoints. In addition, endpoint authentication of middleboxes and vice-versa may be better conducted out-of- band (treating the middlebox as an endpoint for the authentication protocol) than in-band (treating the middlebox as a participant in a 3+ party communication).

7.4. Return routability and feedback

We have identified a requirement to support as wide a range of overlying transports as possible and feasible, in order to maximize SPUD's potential for improving the evolvability of the transport stack.

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 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. In addition, to defend against state exhaustion attacks on middleboxes, SPUD may need to see a first packet in a reverse direction on a tube to consider that tube acknowledged and valid.

Return routability is therefore a minimal property of any transport that can be responsibly deployed at scale in the Internet. Therefore SPUD should enforce bidirectional communication at start-up, whether the overlying transport is bidirectional or not. This excludes use of the UDP source port as an entropy input that does not accept traffic (i.e., for one-way communication, as is commonly done for unidirectional UDP tunnels, e.g., MPLS in UDP [RFC7510]).

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

Discussions about SPUD to date have focused on the possibility of in-band signaling from endpoints to middleboxes and back - the signaling channel happens on the same 5-tuple as the data carried by the overlying transport. This arrangement would have 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 overlying transport (and telling the overlying transport about this MTU reduction, or relying on the overlying transport to use PLPMTUD [RFC4821]), 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 any case, a SPUD sender would effectively request SPUD information

from a middlebox that respectively the middlebox would be able to insert the requested information into this place holder.

However, a SPUD using piggybacked signaling is at the mercy of the overlying transport's transmission scheduler to actually decide when to send packets. At the same time, piggybacked signaling has the

benefit that SPUD can use the overlying transport's reliability mechanisms to meet any reliability requirements it has for its own use (e.g. in key exchange). This would require SPUD to understand the semantics of the overlying protocol but can reduce overhead. Piggyback signaling is also the only way to achieve per-packet signaling as in Section 7.2.

Interleaved signaling uses SPUD-only packets on the same 5-tuple with the same tube identifier to achieve signaling. This reduces complexity and sidesteps MTU problems, but is only applicable to per-tube signaling. It also would require SPUD to provide its own reliability mechanisms if per-tube signaling requires reliability, and still needs to interact well with the overlying transport's transmission scheduler.

Out-of-band signaling uses direct connections between endpoints and middleboxes, separate from the overlying 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 overlying transport and 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 the final approach may use a mix of all these communication patterns where SPUD provides different signaling patterns for different use case. E.g., a middlebox might need to generate out-of-band signals for error messages or can provide requested information in-band and feedback over the receiver if a minimum or maximum value from all SPUD-aware middleboxes on path should be discovered.

7.6. Continuum of trust among endpoints and middleboxes

There are different security considerations for different security contexts. The end-to-end context is one; anything that only needs to

be seen by the path shouldn't be exposed in SPUD, but rather by the overlying transport. There are multiple different types of end-to-middle context based on levels of trust between end and middle - is the middlebox on the same network as the endpoint, under control of

the same owner? Is there some contract between the application user and the middlebox operator? It may make sense for SPUD to support different levels of trust than the default ("untrusted, but presumed honest due to limitations on the signaling vocabulary") and fully-authenticated; this needs to be explored further.

7.7. Discovery and capability exposure

There are three open issues in discovery and capability exposure. First, an endpoint needs to discover if the other communication endpoint understands SPUD. Second, endpoints need to test whether SPUD is potentially not usable along a path because of middleboxes that block SPUD packets or strip the SPUD header. If such impairments exist in the path, a SPUD sender needs to fall back to some other approach to achieve the goals of the overlying transport. Third, endpoints might want to be able to discover SPUD-aware middleboxes along the path, and to discover which parts of the vocabulary that can be spoken by the endpoints are supported by those middleboxes as well as the other communication endpoint, and vice versa.

In addition, endpoints may need to discover and negotiate which overlying transports are available for a given interaction. SPUD could assist here. However, it is explicitly not a goal of SPUD to expose information about the details of the overlying transport to middleboxes.

7.8. Hard state vs. soft state

The initial thinking on signaling envisions "hard state" in middleboxes that is established when the middlbox observes the start of a SPUD tube and is torn down when the middlebox observes the end (stop) of a SPUD tube. Such state can be abandoned as a result of network topology changes (e.g., routing update in response to link or node failure). An alternative is a "soft state" approach that requires periodic refresh of state in middleboxes, but cleanly times out and discards abandoned state. SPUD has the opportunity to use different timeouts than the defaults that are required for current NAT and firewall pinhole maintenance. Of course, applications will still have to detect non-SPUD middleboxes that use shorter timers.

8. Security Considerations

The security-relevant requirements for SPUD deal mainly with endpoint authentication and the integrity of exposed information (Section 5.5, Section 5.6, Section 5.7, and Section 7.3); protection against attacks (Section 7.1 and Section 7.4); and the trust relationships among endpoints and middleboxes Section 7.6. These will be further

addressed in protocol definition work following from these requirements.

9. IANA Considerations

This document has no actions for IANA.

10. Contributors

In addition to the editors, this document is the work of David Black, Ken Calvert, Ted Hardie, Joe Hildebrand, Jana Iyengar, and Eric Rescorla.

[EDITOR'S NOTE: make this a real contributor's section once we figure out how to make kramdown do that...]

11. Informative References

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, September 1981.

[RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, May 2000.

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

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

[RFC7510] Xu, X., Sheth, N., Yong, L., Callon, R., and D. Black, "Encapsulating MPLS in UDP", RFC 7510, April 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.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-00 (work in progress), June 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.

Authors' Addresses

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

Email: ietf@trammell.ch

Mirja Kuehlewind (editor)
ETH Zurich
Gloriastrasse 35
8092 Zurich
Switzerland

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