Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 21, 2016

B. Trammell, Ed.
M. Kuehlewind, Ed.
ETH Zurich

October 19, 2015

Requirements for the design of a Substrate Protocol for User Datagrams (SPUD)
draft-trammell-spud-req-01

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.

Table of Contents

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 possibly chartering a 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. 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. 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.

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 [stackevo-explicit-coop]. 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:

- Superstrate: : The transport protocol or protocol stack "above" SPUD, that uses SPUD for explicit path cooperation and path traversal. The superstrate usually consists of a security layer (e.g. TLS, DTLS) and a transport protocol, or a transport protocol with integrated security features, to protect headers and payload above SPUD.

- Endpoint: : One end of a communication session, located on a single node that is a source or destination of packets in that session. In this document, this term may refer to either the SPUD implementation at the endpoint, the superstrate implementation running over SPUD, or the applications running over that superstrate.

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

- Middlebox: : As defined in [RFC3234], a middlebox is any intermediary device performing functions other than the normal, standard functions of an IP router on the datagram path between a source host and destination host; e.g. making decisions about forwarding behavior based on other than addressing information, and/or modifying a 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, ack, and stop) to a flow or a group of packets within a flow which are semantically related in terms of the application or superstrate. 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" for error messages which should be application-visible; these would traverse NATs in the same way as the traffic related to it, and be deliverable to the application with appropriate tube information.

5. Functional Requirements

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

5.1. Grouping of Packets (into "tubes")

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. Each packet in a SPUD "flow" (determined by 5-tuple) belongs to exactly one tube. Notionally, a tube consists of a set of packets with a set of common properties, that should

therefore receive equivalent treatment from the network; these tubes may or may not be related to separate semantic entities in the superstrate (e.g. SCTP streams).

The simplest mechanisms for association involve the addition of an identifier to each packet in a tube. Current thoughts on the tradeoffs on requirements and constraints on this identifier space are given in {{tradeoffs-in-tube- identifiers}}.

5.2. Endpoint to Path Signaling

SPUD must be able to provide information scoped to a tube from the end- point(s) to all SPUD-aware nodes on the path about the packets in that tube. Since it is implausible that an endpoint has pre-existing trust 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 appropriate. See {{in-band-out-of- band-piggybacked-and-interleaved-signaling}} for more discussion.

5.3. Path to Endpoint Signaling

SPUD must be able to provide information from a SPUD-aware middlebox to the endpoint. Though this information is not scoped to a tube in the same way that endpoint to path signaling is, as the middleboxes do not originate the packets in a tube, it is still associated with a tube, in terms of "the properties of the path(s) this tube will traverse". Path to endpoint signaling need not be in-band; see Section 7.4 for more discussion.

5.4. Tube Start and End Signaling

SPUD must provide a facility for endpoints to signal that a tube has started, that the start of the tube has been acknowledged and accepted by the remote endpoint(s), and that a tube has ended and its state can be forgotten by the path. Given unreliable signaling (see Section 6.11) both endpoints and devices on the path must be resilient to the loss of any of these signals. Specifically, timeouts are still necessary to clean up stale state. See Section 7.7 and Section 7.8 for more discussion on tube start and end signaling.

5.5. Extensibility

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

5.6. Authentication

The basic SPUD protocol must not require any authentication or a priori trust relationship between endpoints and middleboxes to function. However, SPUD should interoperate with 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, and use this information where possible and appropriate.

5.7. Proof a device is on-path

Devices may make assertions of network characteristics relevant to a flow. One way these assertions can be assessed is by a demonstration that the device making it is on-path to the flow and so could adjust the characteristics to match the assertion. SPUD must therefore allow endpoints to distinguish on- path devices from devices not on the path. Network elements may also need to confirm that application-to-path assertions are made by the source indicated in the flow. In both cases, return routability (as in {{protection-against- trivial-abuse}}) may offer one incrementally deployable method of testing the topology to make this confirmation.

5.8. Integrity

SPUD must provide integrity protection of SPUD-encapsulated packets, though the details of this integrity protection are still open; see {{tradeoffs-in- integrity-protection}}. Endpoints should be able to detect changes to headers SPUD uses for its own signaling (whether due to error, accidental modification, or malicious modification), as well as the injection of packets into a SPUD flow (defined by 5-tuple) or tube by nodes other than the remote endpoint. Integrity protection of the superstrate is left up to the superstrate.

5.9. 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. Technical 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. This encapsulation will require port numbers to support NAPT- connected endpoints. UDP encapsulation is the only mechanism that meets these requirements.

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 superstrates 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. This indicates 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 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 {{stackevo- explicit-coop}} for more.

6.5. Protection against trivial abuse

Malicious background traffic is a serious problem for UDP- based protocols due to the ease of forging source addresses in UDP together with the only limited deployment of network egress filtering

[RFC2827]. Trivial abuse includes flooding and state exhaustion attacks, as well as reflection and amplification attacks. SPUD must provide minimal protection against this trivial abuse. This probably implies that SPUD should provide:

- a proof of return routability,
- a feedback channel between endpoints,
- a method to probabilistically discriminiate legitimate SPUD traffic from reflected malicious traffic, and
- mechanisms to protect against state exhaustion and other denial-of-service attacks.

We note that return routability excludes use of a UDP source port 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] as an entropy input.)

6.6. No unnecessary restrictions on the superstrate

Beyond those restrictions deemed necessary as common features of any secure, responsible transport protocol (see Section 6.5), SPUD must impose only 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 superstrate.

6.7. Minimal additional start-up latency

SPUD should not introduce additional start-up latency for superstrates.

6.8. Minimal 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.

6.9. Minimal non-productive traffic

SPUD should minimize additional non-productive traffic (e.g. keepalives), and should provide mechanisms to allow its superstrates to minimize their reliance on non-productive traffic.

6.10. Preservation of Security Properties

The use of SPUD must not weaken the security properties of the superstrate. If the superstrate includes payload encryption for confidentiality, for example, the use of SPUD must not allow deep packet inspection systems to have access to the plaintext. While a box along the path may indicate a particular flow is adminstratively prohibited or why it is prohibited, SPUD itself must not be used to negotiate the means to lift the prohibition.

6.11. Reliability, Fragmentation, and Duplication

As any information provided by SPUD is anyway opportunistic, SPUD need not provide reliable signaling for the information associated with a tube. Signals must be idempotent; all middleboxes and endpoints must gracefully handle receiving duplicate signal information. To avoid issues with fragment reassembly, all in-band SPUD signaling information must fit within a single packet. Any facilities requiring more than an MTU's worth of data in a single signal should use an out-of-band method which does provide reliability - this method may be an existing transport or superstrate/SPUD combination, or a "minimal transport" defined by SPUD for its own use.

6.12. Interoperability with non-encapsulated superstrates

It is presumed that "superstrate X with SPUD" is a distinct entity on the wire from "superstrate X". The APIs the superstrate presents to the application should be equivalent, and the two wire protocols should be freely transcodeable between each other, with the caveat that the variant without SPUD would not necessarily support features enabling communication with the path. However, there is no requirement that the headers the superstrate uses be the same in the SPUD and non-SPUD variants. Headers that the superstrate chooses always to expose to the path can therefore be encoded in the SPUD layer but not appear in an upper-layer header.

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 superstrates. 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 minimize the probability of collision among multiple tubes having the same identifier along the same path during some period of time. A 128-bit UUID [RFC4122] or an identifier generated using an equivalent algorithm would be useful as such a globally- unique tube identifier. An advantage of globally unique tube identifiers would be 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. This alone probably speaks against using globally unique identifiers for SPUD.

In the case of 5-tuple-scoped identifiers, mobility must be supported separately from the tube identification mechanism. This could be specific to each superstrate (i.e., hidden from the path), or SPUD could provide a general endpoint-to-path tube grouping signal to allow an endpoint to explicitly expose the fact that one tube is related to another to the path. Even in this case, 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.

When scoped to 5-tuples, the forward and backward directions of a bidirectional flow probably have different tube IDs, since these will necessarily take different paths and may interact with a different set of middleboxes due to asymmetric routing. SPUD will therefore require some facility to note that one tube is the "reverse"

direction of another, a general case of the tube grouping signal above.

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. It is also possible that SPUD will provide a very limited set of per-packet signals (such as ECN) using flags in the SPUD header, and require all more complicated properties to be bound per- tube.

7.3. Tradeoffs in integrity protection

In order to protect the integrity of information carried by SPUD against forging by malicious devices along the path, it would be 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 superstrate; in this case, SPUD may be able 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. 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 superstrate. However, there are a wide variety of potential signaling arrangements: in-band signaling can be piggybacked (where signaling happens on packets sent by the superstrate) and/or interleaved (where SPUD and the superstrate each have their own packets). Signaling can also be out-of-band (on a different five tuple, or even over a completely different protocol). Out of band signaling for path-to-endpoint information can use direct return,

allowing a device on the path to communicate directly with an endpoint (i.e., as with ICMP). More discussion on the tradeoffs here is given in [stackevo-explicit-coop].

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 situations. 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.

7.5. 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 superstrate. 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? SPUD should support different levels of trust than the default ("untrusted, but presumed honest due to limitations on the signaling vocabulary") and fully-authenticated; how these points along the continuum are to be implemented and how they relate to each other needs to be explored further.

7.6. 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 superstrate. 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 superstrates 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 superstrate to middleboxes.

7.7. Hard state vs. soft state

The initial thinking on signaling envisions "hard state" in middleboxes that is established when the middlebox 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.

7.8. Tube vs. superstrate association lifetime

The requirements as presently defined use tube start and stop signaling for two things: (1) setting up and tearing down state along the path, and (2) signaling superstrate such as association startup, acceptance, and teardown, which may have security implications. These may require separate signaling. Specifically, if tube start acknowledgement is to be used to provide explicit guarantees to the path about the acceptability of a tube to a remote endpoint, it cannot be a completely unreliable signal. Second, the lifetime of a tube may be much shorter than the lifetime of a superstrate association, and the creation of a new tube over an existing association may need to be treated differently by endpoints and path devices than a tube creation coincident with an association creation.

8. Security Considerations

The security-relevant requirements for SPUD deal mainly with endpoint authentication and the integrity of exposed information (Section 5.6, Section 5.8, Section 5.9, and Section 7.3); protection against attacks (Section 5.7, Section 6.5, and Section 7.1 and); and the trust relationships among endpoints and middleboxes Section 7.5. 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.

11. Acknowledgments

Thanks to Roland Bless, Cameron Byrne, Toerless Eckert, Daniel Kahn Gillmor, Tom Herbert, and Christian Huitema for feedback and comments on these requirements, as well as to the participants at the SPUD BoF at IETF 92 meeting in Dallas inand the IAB SEMI workshop in Zurich for the discussions leading to this work.

12. Informative References

[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>.

[RFC3234] Carpenter, B. and S. Brim, "Middleboxes: Taxonomy and Issues", RFC 3234, DOI 10.17487/RFC3234, February 2002, <http://www.rfc-editor.org/info/rfc3234>.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <http://www.rfc-editor.org/info/rfc4122>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <http://www.rfc-editor.org/info/rfc6347>.

[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.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.

[stackevo-explicit-coop]
Trammell, B., "Architectural Considerations for Transport Evolution with Explicit Path Cooperation", September 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.

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