Workgroup: TODO Working Group
Internet-Draft:
draft-krose-multicast-security-03
Published: 11 July 2022
Intended Status: Standards Track
Expires: 12 January 2023
K. Rose
Akamai Technologies, Inc.
J. Holland
Akamai Technologies, Inc.

Security and Privacy Considerations for Multicast Transports

Abstract

Interdomain multicast has unique potential to solve delivery scalability for popular content, but it carries a set of security and privacy issues that differ from those in unicast delivery. This document analyzes the security threats unique to multicast-based delivery for Internet and Web traffic under the Internet and Web threat models.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the mailing list (), which is archived at .

Source for this draft and an issue tracker can be found at https://github.com/squarooticus/draft-krose-multicast-security.

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 https://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 12 January 2023.

Copyright Notice

Table of Contents

1. Introduction

This document examines the security considerations relevant to the use of multicast for scalable one-to-many delivery of application traffic over the Internet, along with special considerations for multicast delivery to clients constrained by the Web security model.

1.1. Background

This document assumes readers have a basic understanding of some background topics, specifically:

*The Internet threat model as defined in Section 3 of [RFC3552].

*The Security Considerations for UDP Usage Guidelines as described in Section 6 of [RFC8085], since application layer multicast traffic is generally carried over UDP.

*Source-specific multicast, as described in [RFC4607]. This document focuses on interdomain multicast, therefore any-source multicast is out of scope in accordance with the deprecation of interdomain any-source multicast in [RFC8815].

1.2. Web Security Model

The Web security model, while not yet documented authoritatively in a single reference, nevertheless strongly influences Web client implementations, and has generally been interpreted to require certain properties of underlying transports such as:

*Confidentiality: A passive observer must not be able to identify or access content through simple observation of the bits being delivered, up to the limits of metadata privacy (such as traffic analysis, peer identity, application/transport/security-layer protocol design constraints, etc.).

*Authenticity: A receiver must be able to cryptographically verify that the delivered content originated from the desired source.

*Integrity: A receiver must be able to distinguish between original content as sent from the desired source and content modified in some way (including through deletion) by an attacker.

*Non-linkability: A passive observer must not be able to link a single user across multiple devices or a single client roaming across multiple networks.

For unicast transport, TLS [RFC8446] satisfies these requirements, therefore Web Transport [webtrans] proposes to require qualifying transport protocols to use "TLS or a semantically equivalent security protocol".

For unicast communication this is sensible and meaningful (if imprecise) for an engineer with a grounding in security, but it is unclear how or whether 'semantic equivalence to TLS' can be directly interpreted in any meaningful way for multicast transport protocols. This document instead explicitly describes a security and privacy threat model for multicast transports in order to extend the Web security model to accommodate multicast delivery in a way that fits within the spirit of how that model is generally interpreted for unicast.

Although defining the security protections necessary to make multicast traffic suitable for Web Transport is a key goal for this document, many of the security considerations described here would be equally necessary to consider if a higher level multicast transport protocol were to be made available via a different interface within clients constrained by the Web security model.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Threat Model

Fundamentally, multicast is simply an addressing scheme in which the destination address identifies more than one unique receiver; that said, this has implications for protocol design that differ greatly from those for unicast addressing.

Given the virtually unbounded potential for attacks targeting data confidentiality and user privacy, we attempt to make the description of a multicast threat model tractable by taking the approach of highlighting areas in which multicast differs from unicast or poses novel challenges that are not addressed at a layer unconcerned with the addressing scheme.

3.1. Multicast Transport Properties

Unlike typical unicast transport protocols, multicast transports are naturally unidirectional. Use cases for multicast transports typically involve one or a small number of senders transmitting data to a large number of receivers. The sender may not know who the receivers are, or even how many of them there are, although a sender may require a pre-existing out-of-band relationship with receivers for the received data to be useful, such as via distribution of decryption keys only to authorized receivers.

Applications built atop multicast IP or UDP must provide a mechanism for congestion control, just as those built atop unicast IP or UDP must. Although multicast applications compliant with Section 4.1 of [RFC8085] will implement congestion control, in the context of a threat model it is important to note that malicious clients might attempt to use non-compliant subscriptions to multicast traffic as part of a DoS attack where possible, and that some applications might not be compliant with the recommendations for congestion control implementations.

IP and UDP provide no native reliability mechanism, whether for unicast or multicast transmission. Protocols leveraging multicast may add mechanisms for reliable delivery (see [RFC5740], [RFC5775], and [quic-http-mcast] for examples), but this may expand the attack surface against content providers if per-packet authenticity is not provided. For example, in an application with unicast recovery for objects constructed out of multiple packets and which is limited to object-level authentication, if a packet is injected into the multicast stream receivers will fail to authenticate an entire object, necessitating unicast recovery by every receiver for the entire object. Care must be taken to avoid such amplification attack vectors.

3.2. Authentication

The Web security model requires that data delivered to applications must be authenticated as having originated from the trusted peer. (In the case of server-only transport-level authentication schemes, such as the ubiquitous TLS server-only authentication employed throughout the Web, trust in the client may be strongly established at the application layer or weakly established as nothing more than precluding a man-in-the-middle.)

In the unicast case, authentication of payloads in HTTPS is provided by a trusted octet stream, cryptographically resistant to tampering, bootstrapped via certificate validation and trust chain verification (either mutual or server-only, perhaps augmented with application-layer identity verification).

Authentication of units of transport for trusted channels (think: individual packets or messages) is typically provided by a cryptographic authentication tag:

- Symmetric tags, such as symmetric message authentication codes (MACs) and authentication tags produced by authenticated encryption (AE) algorithms. Because anyone in possession of the keying material may produce valid symmetric authentication tags, such keying material is typically known to at most two parties: one sender and one receiver. Some algorithms employing symmetric authentication (such as TESLA, discussed below) relieve this constraint by imposing some different constraint on verification of tagged content.

- Asymmetric tags, typically signatures produced by public key cryptosystems. These assume that only the sender has access to the signing key, but impose no constraints on dissemination of the signature verification key.

In both cases:

- The receiving party must have a means for establishing trust in the keying material used to verify the authentication tag.

- Instead of directly authenticating the protected content, the tag may protect a root of trust that itself protects cryptographically-linked content. Examples include:

  - The TLS 1.3 handshake employing an authentication tag to reject MitM attacks against ECDH key agreement.

  - An authentication tag of a Merkle tree root protecting the content represented by the entire tree.

- The authentication tag serves to provide integrity protection over the unit of content to which the tag applies, with additional mechanisms required to detect and/or manage duplication/replay, deletion/loss, and reordering within a sequence of such authenticated content units.

Asymmetric verification of content delivered through multicast is conceptually identical to the unicast case, owing to the asymmetry of access to the signing key; but the symmetric case does not directly apply given that multiple receivers need access to the same key used for both signing and verification, which in a naive implementation opens up the possibility of forgery by a receiver on-path or with the ability to spoof the source.

Multiple mechanisms providing for reliable asymmetric authentication of data delivered by multicast have been proposed over the years.

- TESLA [RFC4082] achieves asymmetry between the sender and multiple receivers through timed release of symmetric keying material rather than through the assumed computational difficulty of deriving a signing key from a verification key in public key cryptosystems like RSA and ECDSA. It employs computationally-inexpensive symmetric authentication tagging with release of the keying material to receivers only after they are assumed to have received the protected data, with any data received subsequent to scheduled key release to be discarded by the receiver. This requires some degree of time synchronization between clients and servers and imposes latency above one-way path delay prior to release of authenticated data to applications.

- Simple per-packet asymmetric signature of packet contents based on out-of-band communication of the signature's public key and algorithm, for example as described in Section 3 of [RFC6584].

- Asymmetric Manifest-based Integrity (AMBI) [AMBI] relies on an out-of-band authenticated channel for distribution of manifests containing cryptographic digests of the packets in the multicast stream. Authentication of this channel may, for instance, be provided by TLS if manifests are distributed using HTTPS from an origin known to the client to be closely affiliated with the multicast stream, such as would be the case if the manifest URL is delivered by the origin of the parent page hosting the media object. Authenticity in this case is a prerequisite of the out-of-band channel that AMBI builds upon to provide authenticity for the multicast data channel.

Regardless of mechanism, the primary goal of authentication in the multicast context is identical to that for unicast: that the content delivered to the application originated from the trusted source. Semantic equivalence to (D)TLS in this respect is therefore straightforwardly achieved by any number of potential mechanisms.

3.3. Integrity

Integrity in the Web security model for unicast is closely tied to the features provided by transports that enabled the Web from its earliest days. TCP, the transport substrate for the original HTTP, provides in-order delivery, reliability via retransmission, packet de-duplication, and modest protection against replay and forgery by certain classes of adversaries. SSL and TLS later greatly strengthened those protections. Web applications universally rely on these integrity assumptions for even the most basic operations. It is no surprise, then, that when QUIC was subsequently designed with HTTP

as the model application, initial requirements included the integrity guarantees provided by TCP at the granularity of an individual stream.

Multicast applications by contrast have different integrity assumptions owing to the multicast transport legacy. UDP, the transport protocol atop which multicast applications are typically built, provides no native reliability, in-order delivery, de-duplication, or protection against replay or forgery. Additionally, UDP by itself provides no protection against off-path spoofing or injection. Multicast has therefore traditionally been used for applications that can deal with a modest loss of integrity through application-layer mitigations such as:

- Packet indexes to reveal duplication/replay and reordering, and to complicate off-path spoofing and injection
- Deletion coding to allow for passive recovery from loss/deletion
- Graceful degradation in response to loss/deletion, exemplified by video codecs designed to tolerate loss

A baseline for multicast transport integrity that makes sense within the Web security model requires that we first define the minimally acceptable integrity requirements for data that may be presented to a user or otherwise input to the browser's trusted computing base. We propose that the proper minimal standard given the variety of potential use cases, including many that have no need for reliable or in-order delivery, is to require protection against replay, injection, and modification and the ability to detect deletion, loss, or reordering. This standard will necessarily constrain conformant application-layer protocol design, just as the Web security model adds constraints to vanilla TCP.

Integrity in multicast, as in the unicast case, is partially provided by the authentication mechanism: for example, if authentication is provided at packet granularity, modified or forged packets will fail to authenticate and will thus not be delivered to the application. Lacking a bidirectional relationship at the transport layer, however, applications relying on multicast must otherwise provide for detection of and/or recovery from packet duplication/replay, loss/deletion, and reordering. Some of these functions, too, may be provided by the authentication layer. For instance:

- TESLA prevents replay and reveals reordering, but only across time intervals. An application requiring finer-grained countermeasures against duplication/replay or reordering, or indeed any countermeasure to deletion/loss, would need to provide that via custom support (e.g., through the introduction of packet sequence numbers) or via an intermediate-layer protocol providing those functions.
- AMBI by design provides strong protection against duplication/replay and reveals reordering and deletion/loss of content packets through a strict in-order manifest of packet digests.

3.4. Confidentiality

In the unicast transport security context, confidentiality implies that an observer (passive or active) without pre-existing access to keying material must not be able to decrypt the bytes on the wire or identify the content being transferred, even if that adversary has access to the decrypted content via other means. In practice, the former is trivially achieved through the use of authenticated key exchange and modern symmetric ciphers, but the latter is an ideal that is rarely possible owing to the substantial metadata in the clear on the public Internet: traffic analysis can make use of packet sizes and timing, endpoint identities, biases in application-layer protocol designs, side channels, and other such metadata to reveal an often surprising amount of information about the encrypted payload without needing access to any keying material. (Conceptually, one could make many streams appear identical to a passive observer: video streams, for example, could be bucketed into a small number of bitrates with identical packet sizes and pacing via padding of the actual content. This would increase overhead for servers and networks, primarily in terms of bandwidth utilization, that may be operationally unacceptable.)

Multicast additionally introduces the complication that all receivers of a stream, even if such a stream is encrypted, receive the same payload (loss and duplication notwithstanding). This introduces novel privacy concerns that do not apply to unicast transports.

3.4.1. Privacy

In contrast to (say) unicast TLS, on-path monitoring can trivially prove that identical content was delivered to multiple receivers, irrespective of payload encryption. Furthermore, since those receivers all require the same keying material to decrypt the received payload, a compromise of any single receiver's device exposes decryption keys, and therefore the plaintext content, to the attacker.

That having been said, however, there are factors and practices that help mitigate these additional risks:

* Multicast delivery is unidirectional from content provider to consumer and has no end-to-end unicast control channel association at the transport-layer, though such associations are generally unavoidable at the application layer (a common case likely being a referring web page). Assuming application-layer unicast control plane traffic is properly secured, identifiable plaintext control messages are limited to IGMP or MLD messages intercepted by (and not retransmitted with user-identifying information by) a user's upstream router.

  Notwithstanding linkability via data or metadata from application-layer control flows, an on-path observer can thus only directly determine that some entity downstream of that path element has joined a particular multicast channel (in SSM [RFC4607], identified by the (source, group) pair of IP addresses). Lacking a destination address, increasing the specificity of receiver identification would require an

observer to obtain monitoring points closer to the user or to manipulate a user into revealing metadata out-of-band that the observer can tie to the user via traffic analysis or other means.

This is a form of k-anonymity not available to unicast transports. In the unicast case, an on-path observer has access to metadata specific to endpoint address pairs, including total flow size, packet count, port and protocol, which (in combination with other metadata) can later be tied to the user, site, service, and/or location assigned to each address at the given time.

Widespread near-simultaneous unicast download events, such as those triggered by the release of a video game update or of an episode of a popular streaming video series, expose the identities of consumers of such content anywhere along the path from end users' devices to the origin through very elementary traffic analysis, unless measures are taken by the end user or content provider to hide the traffic, such as by mixing it with other traffic in a way that complicates disentangling individual flows. A properly-designed virtual private network (VPN) link could, for example, obfuscate flow-identifying information in traffic to a given user, at the expense of using greater bandwidth (for added chaff) and of loudly signaling to passive observers the presence of a VPN link.

*There is no standard mechanism in the multicast protocol ecosystem by which a passive observer may derive separate but related content or metadata from the multicast channel itself: in particular, if a multicast stream is encrypted using a key delivered out-of-band, there is no general means by which a passive observer could directly derive the source location of the keying material. For a passive observer to know what encrypted content is being delivered to a particular user whose channel subscriptions are known they would need to already know what content is available via that channel, either via traffic analysis such as in the case of passive observation of unicast TLS, or via a priori knowledge of related content that references the channel. A dragnet cataloging all content available through a particular origin is an example of the latter, but could be further mitigated via controlled access to index information, or via periodic changes in multicast source, group, or keying material, or some combination of the three.

3.4.2. Personal Data

A sender has responsibility not to expose personal information broadly. This is not a consideration unique to multicast delivery: an irresponsible service could publish a web page with Social Security numbers or push its server TLS private key into the certificate transparency log as easily as it could multicast personal data to a large set of receivers.

The Web security model partially mitigates negligence on the part of senders by mandating the use of secure transports: prohibiting the fetching of mixed content on a single page prevents a server from

sending private data to a browser in the clear. The main effect is to raise the bar closer to requiring bad faith or willful irresponsibility on the part of senders in revealing personal information.

Multicast by its very nature is not generally suitable for transport of personal data: since the main value of leveraging a multicast transport is to deliver the same data to a large pool of receivers, such content must not include confidential personal information. Senders already have a responsibility to handle private information in a way that respects the privacy of users: the availability of multicast transports does not further complicate this responsibility.

3.4.3. Forward Secrecy

Forward secrecy (also called "perfect forward secrecy" or "PFS" and defined in [RFC4949]) is a countermeasure against attackers that record encrypted traffic with the intent of later decrypting it should the communicating parties' long-term keys be compromised. Forward secrecy for protocols leveraging time-limited keys to protect a communication session ("session keys") requires that such session keys be unrecoverable by an attacker that later compromises the long-term keys used to negotiate or deliver those session keys.

As noted earlier, confidential content delivered via multicast will necessarily imply delivery of the same keying material to multiple receivers, rather than negotiation of a unique key as is typical in the unicast case. Presumably, such receivers will need to be individually authenticated and authorized by the content provider prior to delivery of decryption keys. If this authorization and key delivery mechanism employs a forward secret unicast transport such as TLS 1.3, then so long as these encryption keys are ephemeral (that is, rotated periodically and discarded after rotation) the multicast payloads will also effectively be forward secret beyond the time interval of rotation, which we can consider to be the session duration.

3.4.4. Bypassing Authentication

Protocols should be designed to discourage implementations from making use of unauthenticated data. The usual approach to enforcing this is to entangle decryption and authentication where possible, for example via use of primitives such as authenticated encryption. While ultimately authentication checks are independent of decryption (at least in classical cryptography), use of such primitives to minimize the number of places in which an incomplete or lazy implementation can avoid such checks constitutes best practice. TLS 1.3, for instance, mandates AE for all symmetric cryptographic operations: without writing one's own AE cipher implementation that purposely skips the authentication tag check, this leaves establishment of trust in the peer certificate as the only practical step an implementation can skip without impacting the ability to make use of the decrypted content.

The situation in multicast is complicated by the need for more than two parties to have access to symmetric keys that would used to secure payloads via AE in the unicast case. As discussed in Section

3.2, it is imperative for protocols to provide, and for receivers to leverage, some kind of asymmetry in authentication of each content unit prior to any use of said content to eliminate the ability for an attacker in possession of a shared symmetric key (possibly including an authorized receiver) to inject forged data into a stream that other receivers would then validate and deliver to applications. This requirement to perform authentication checks throughout the lifetime of a stream that are separate from, and orthogonal to, content decryption adds an extra dimension of risk from implementation incorrectness, because such authentication becomes an on-going process rather than the result of a one-time certificate check at connection establishment. Protocol designers and implementors are thus strongly encouraged to simplify or even black box such on-going authentication to minimize the potential for implementors or users to skip such checks.

3.5. Request/Response Binding

In addition to requiring that application data be cryptographically authenticated, the Web security model also requires that each HTTP response must be bound to a specific HTTP request in a way that cannot be forged by an adversary.

In the unicast case, binding of a response to a request in HTTPS is a direct consequence of the integrity guaranteed by the cryptographically protected transport stream to the image of the underlying HTTP protocol, which itself allows for no ambiguity in determining which request induced a particular response.

Request/response binding in multicast requires additional protocol or application-layer support as multicast is naturally unidirectional and so does not carry request traffic. Any multicast protocol carrying Web traffic must provide a means for cryptographically binding the data delivered over a multicast channel to a specific client request. Failure to do so could, for example, allow an on-path adversary to swap the packets between two different multicast channels both trusted by the client without being detected prior to delivery to the application.

3.6. Non-linkability

Concern about pervasive monitoring of users culminated in the publication of [RFC7258], which states that "the IETF will work to mitigate the technical aspects of [pervasive monitoring]." One area of particular concern is the ability for pervasive monitoring to track individual clients across changes in network connectivity, such as being able to tell when a device or connection migrates from a wired home network to a cell network. This has motivated mitigations in subsequent protocol designs, such as those discussed in section 9.5 of [RFC9000]. Migration of multicast channel subscriptions across network connections carries the potential for correlation of metadata between multicast channel subscriptions and unicast control channels, even when control channels are encrypted, so care must be taken to design protocols to avoid such correlations.

3.7. Browser-Specific Threats

The security requirements for multicast transport to a browser follow directly from the requirement that the browser's job is to protect the user. Huang et al. [huang-w2sp] summarize the core browser security guarantee as follows:

*Users can safely visit arbitrary web sites and execute scripts provided by those sites.

The reader will find the full discussion of the browser threat model in section 3 of [RFC8826] helpful in understanding what follows.

3.7.1. Access to Local Resources

This document covers only unidirectional multicast from a server to (potentially many) clients, as well as associated control channels used to manage that communication and access to the content delivered via multicast. As a result, local resource access can be presumed to be limited to that already available within web applications. Note that these resources may include fingerprint information that can be used to identify or track individuals, such as information about the user agent, viewport size, display resolution, a concern covered in extensive detail in [RFC8942].

3.7.2. Injection

In the absence of any specific mitigations, network attackers have the ability to inject or modify packets in a multicast stream. On-path injection and modification are trivial, but even off-path injection is feasible for many networks, such as those that implement no protections against source address spoofing. Consequently, it is critical that a browser prevent any such injected or modified traffic from reaching large attack surfaces in the browser, such as the rendering code.

3.7.3. Hostile Origin

A hostile origin could serve a Web application that attempts to join many multicast channels, overwhelming the provider's network with undesired traffic.

The first line of defense is the browser itself: the browser should at a minimum prevent joining of channels not associated with the hosting site. In the general case, this implies the need for a CORS-like mechanism for cross-origin authorization of multicast channel sharing.

The second line of defense is the network. The user's upstream router can and should monitor the user's multicast behavior, implementing circuit breakers that will target unpopular content when overloaded or when an abusive subscription pattern is detected.

3.7.4. Private Browsing Modes

Browsers that offer a private browsing mode, designed both to bypass access to client-side persistent state and to prevent broad classes

of data leakage that can be leveraged by passive and active attackers alike, should require explicit user approval for joining a multicast group given the metadata exposure to network elements of IGMP and MLD messages.

3.8. Other Threats

3.8.1. Referrer Checks

Unicast Web traffic has a weak form of protection against unauthorized use of content by third-party sites through referrer checks. Browsers send a Referer [sic] header containing the parent page URL in requests for objects referenced in that page, which allows the server to reject requests from pages not authorized to refer to such content. This is used, for example, to complicate the hosting of phishing sites, which could otherwise serve only the relatively small page HTML and direct the browser to fetch all other page objects from the legitimate origin. This is not a strong security measure, as clients may render cached versions of such elements without checking freshness with the origin; but it does force the attacker to duplicate, modify, and host more content to convincingly mock the target site.

Protocols enabling the delivery of Web traffic over multicast should include some mechanism providing a similar degree of protection against unauthorized use. This is complicated by the inherently unidirectional nature of multicast traffic, which precludes any active role for the server in preventing data delivery to specific clients. In lieu of this, protocols should be designed in a way that allows properly-functioning clients to unilaterally reject multicast data delivered for objects referenced by pages that the server has not authorized.

4. Security Considerations

This entire document is about security.

5. IANA Considerations

This document has no IANA actions.

6. References

6.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI

10.17487/RFC3552, July 2003, <https://www.rfc-editor.org/rfc/rfc3552>.

[RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, DOI 10.17487/RFC4607, August 2006, <https://www.rfc-editor.org/rfc/rfc4607>.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <https://www.rfc-editor.org/rfc/rfc4949>.

[RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <https://www.rfc-editor.org/rfc/rfc7258>.

[RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <https://www.rfc-editor.org/rfc/rfc8085>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8815] Abrahamsson, M., Chown, T., Giuliano, L., and T. Eckert, "Deprecating Any-Source Multicast (ASM) for Interdomain Multicast", BCP 229, RFC 8815, DOI 10.17487/RFC8815, August 2020, <https://www.rfc-editor.org/rfc/rfc8815>.

6.2. Informative References

[AMBI] Holland, J. and K. Rose, "Asymmetric Manifest Based Integrity", Work in Progress, Internet-Draft, draft-ietf-mboned-ambi-03, 7 March 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-mboned-ambi-03>.

[huang-w2sp] Huang, L-S., Chen, E.Y., Barth, A., Rescorla, E., and C. Jackson, "Talking to Yourself for Fun and Profit", Web 2.0 Security and Privacy (W2SP 2011) , May 2011, <https://ptolemy.berkeley.edu/projects/truststc/pubs/840/websocket.pdf>.

[quic-http-mcast] Pardue, L., Bradbury, R., and S. Hurst, "Hypertext Transfer Protocol (HTTP) over multicast QUIC", Work in Progress, Internet-Draft, draft-pardue-quic-http-mcast-11, 4 July 2022, <https://datatracker.ietf.org/doc/html/draft-pardue-quic-http-mcast-11>.

[RFC4082] Perrig, A., Song, D., Canetti, R., Tygar, J. D., and B. Briscoe, "Timed Efficient Stream Loss-Tolerant Authentication (TESLA): Multicast Source Authentication Transform Introduction", RFC 4082, DOI 10.17487/RFC4082, June 2005, <https://www.rfc-editor.org/rfc/rfc4082>.

[RFC5740] Adamson, B., Bormann, C., Handley, M., and J. Macker, "NACK-Oriented Reliable Multicast (NORM) Transport Protocol", RFC 5740, DOI 10.17487/RFC5740, November 2009, <https://www.rfc-editor.org/rfc/rfc5740>.

[RFC5775] Luby, M., Watson, M., and L. Vicisano, "Asynchronous Layered Coding (ALC) Protocol Instantiation", RFC 5775, DOI 10.17487/RFC5775, April 2010, <https://www.rfc-editor.org/rfc/rfc5775>.

[RFC6584] Roca, V., "Simple Authentication Schemes for the Asynchronous Layered Coding (ALC) and NACK-Oriented Reliable Multicast (NORM) Protocols", RFC 6584, DOI 10.17487/RFC6584, April 2012, <https://www.rfc-editor.org/rfc/rfc6584>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

[RFC8826] Rescorla, E., "Security Considerations for WebRTC", RFC 8826, DOI 10.17487/RFC8826, January 2021, <https://www.rfc-editor.org/rfc/rfc8826>.

[RFC8942] Grigorik, I. and Y. Weiss, "HTTP Client Hints", RFC 8942, DOI 10.17487/RFC8942, February 2021, <https://www.rfc-editor.org/rfc/rfc8942>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

[webtrans] Vasiliev, V., "The WebTransport Protocol Framework", Work in Progress, Internet-Draft, draft-ietf-webtrans-overview-04, 11 July 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-webtrans-overview-04>.

Acknowledgments

TODO acks

Authors' Addresses

Kyle Rose
Akamai Technologies, Inc.
145 Broadway
Cambridge, MA 02144,
United States of America

Email: krose@krose.org

Jake Holland
Akamai Technologies, Inc.
145 Broadway
Cambridge, MA 02144,
United States of America

Email: jakeholland.net@gmail.com