TCPM
Internet-Draft
Intended status: Experimental
Expires: 8 January 2026

M. Piraux
UCLouvain
O. Bonaventure
UCLouvain & WELRI
T. Wirtgen
UCLouvain

7 July 2025

Opportunistic TCP-AO with TLS
draft-piraux-tcp-ao-tls-latest

Abstract

This document specifies an opportunistic mode for TCP-AO. In this mode, the TCP connection starts with a well-known authentication key which is later replaced by a secure key derived from the TLS handshake.

About This Document

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

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-piraux-tcp-ao-tls/.

Discussion of this document takes place on the TCPM Working Group mailing list (mailto:tcpm@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/tcpm/. Subscribe at https://www.ietf.org/mailman/listinfo/tcpm/.

Source for this draft and an issue tracker can be found at https://github.com/IPNetworkingLab/draft-tcp-ao-tls.

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 8 January 2026.

Table of Contents

1. Introduction

The TCP Authentication Option (TCP-AO) [RFC5925] provides integrity protection for long-lived TCP connections. It assumes that the communicating hosts share a Master Key Tuple (MKT). This MKT is used to derive traffic keys to authenticate the TCP packets exchanged by the two hosts. TCP-AO supports different authentication algorithms [RFC5926].

TCP-AO protects the integrity of all the packets exchanged during a TCP connection, including the SYNs. Such a protection is important for some specific services, but many applications would benefit from the integrity protection offered by TCP-AO, notably against RST attacks that can happen later in the connection. Unfortunately, from a deployment viewpoint, for many applications that use long-lived TCP connections, having an existing MKT on the client and the server before establishing a connection is a severe limitation.

This document proposes a way to derive a MKT from the TLS secure handshake [RFC8446]. Before the TLS handshake completes, this document defines default keys which offer a limited protection to the first TCP packets of the connection. These default keys are then replaced by secure keys to protect the integrity of subsequent packets past the TLS handshake. This prevents packet injection attacks that could result in the failure of the TLS connection.

This mechanism can be used to authenticate the TCP packets of BGP sessions when TLS is used as discussed in [CONEXT24].

This document is organised as follows. We provide a brief overview of Opportunistic TCP-AO in section Section 3. Then section Section 4 discusses the required changes to TCP-AO and TLS.

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.

2.1. Notational conventions

This document uses network byte order (that is, big endian) values. Fields are placed starting from the high-order bits of each byte.

3. An overview of Opportunistic TCP-AO

In a nutshell, an opportunistic TCP-AO connection starts like a TCP-AO connection, i.e. the SYNs and all subsequent packets are authenticated, but using a MKT with a default key specified in this document. Then, during the TLS handshake, both endpoints announce the parameters they will use for their MKT. When the TLS handshake completes, they can use their own MKT to protect the TCP packets they send and use their peer MKT to verify the TCP packets they receive. Thus, the beginning of the connection is not protected against packet modifications and packet injection attacks. The real protection only starts once the TLS handshake finishes.

Figure Figure 1 illustrates the establishment of an opportunistic TCP-AO connection. The client sends a SYN packet using the default MKT defined in this document. The TCP-AO option in the SYN packet indicates the use of this default MKT. The server validates the TCP-AO option and replies with an integrity protected SYN+ACK. The client confirms the establishment of the TCP-AO connection with an ACK and sends a TLS ClientHello containing the AO Extension defined in this document. This extension specifies the authentication algorithms that the client will use when sending TCP packets on the connection and whether TCP options will be protected. At this point the server can derive the TLS keys and the TCP-AO keys to use for validating clients packet. The server replies with TLS ServerHello and TLS EncryptedExtensions messages that are sent in packets using the default TCP-AO MKT. To finish the setting up of TCP-AO, the server includes the AO Extension in the sent EncryptedExtensions to announce the parameters it will use to protect the packets it will send. It then installs the new key in its TCP-AO MKT. Upon reception of these messages, the client can derive the TLS and TCP-AO keys. It installs the TCP-AO keys in its MKT and sends the Finished message protected with the new MKT. All the packets exchanged after the Finished message are protected using the MKT derived from the secure TLS handshake.

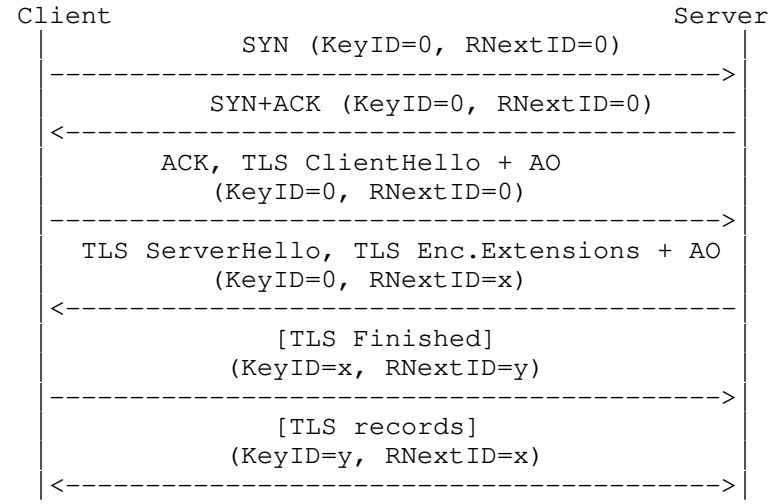

Figure 1: Starting an opportunistic TCP-AO connection with TLS. The messages between brackets are authenticated using the TCP-AO MKT derived from the TLS handshake.

4. Opportunistic TCP-AO

4.1. The TCPAO TLS Extension

This document specifies one TLS extension to support the opportunistic utilization of TCP-AO with keys derived from the TLS secure handshake. The extension is used by endpoints to specify the parameters of the MKT they will use to protect the TCP packets they

send.

```
enum {
    tcp_ao(TBD),
    (65535)
} ExtensionType;
```

The format for the "tcp_ao" extension is defined by:

```
enum {
   tcp_option_protection_disabled(0),
   tcp_option_protection_enabled(1),
   (255)
} TCPAOOptionProt;

enum {
   HMAC-SHA-1-96(0),
   AES-128-CMAC-96(1),
   (255)
} TCPAOAuth;

enum {
   KDF_HMAC_SHA1(0),
   KDF_AES_128_CMAC(1),
   (255)
} TCPAOKDF;

struct {
   TCPAOOptionProt prot;
   TCPAOAuth auth;
   TCPAOKDF kdf;
} TCPAO;
```

The TCPAOOptionProt indicates whether the endpoint will protect the integrity of TCP options or not. The TCPAOAuth specifies the authentication algorithm defined in [RFC5926] that will be used to protect the packets. The TCPAOKDF specifies the key derivation function defined in [RFC5926] and that the endpoint will use to derive its keys.

4.2. The initial MKT

To support the establishment of opportunistics TCP-AO connections, the client and the server must be configured with a default MKT. This default MKT is used to authenticate the packets until the derivation of the secure MKT from the TLS keying material. This document defines the following default MKT:

* TCP connection identifier: selected by the TCP stack.

* TCP option flag. The default MKT assumes that TCP options are not included in the MAC calculation.

* The current values for the SendID and RecvID are set to 0.

* The Master secret is set to 0x1cebb1ff.

* The default key derivation function is KDF_HMAC_SHA1.

* The default message authentication code is HMAC-SHA-1-96.

Given that the TCP-AO KeyID is a local field and has no global meaning, hosts have no guarantee that a KeyID of 0 will be unequivocally recognised as designating the default MKT specified in this document. Section 7.5.1 of [RFC5925] indicates that hosts receiving SYN segments with TCP-AO enabled and no matching MKT should remove the option and accept them. A client initiating a TCP

connection in the opportunistic mode of TCP-AO MUST check that the server accepted the use of TCP-AO in this mode by replying using the default MKT before deriving a secure MKT as described in this document.

4.3. Derivation of the secure TCP AO MKT

The Master key for the MKT to protect the TCP packets after the transmission of the Finished messages are derived from the Exporter Master Secret using Keying Material Exporters [RFC5705]:

```
struct {
   TCPAO ao;
   uint8 key_id;
} TCPAOKeyExporterContext;

TLS-Exporter("tcp-ao", TCPAOKeyExporterContext, 32)
   = tcp_ao_secret
```

The TLS-Exporter function receives the label "tcp-ao", with the parameters of the MKT and the KeyID as context as defined in the TCPAO structure within Section 4.1. It generates a 32-byte secret.

The client and server can decide the value of the KeyID independently and announce it in the AO TCP Option as defined in [RFC5925]. The KeyID MUST be different than the default KeyID of 0.

The traffic keys used by the client and the server can then be derived from this secret using the procedures defined in [RFC5925] and [RFC5926].

After the traffic keys are installed, the client and server stop using the initial MKT defined in Section 4.2.

4.4. Current limitations

This version of the document does not specifiy how to do key updates for MKTs. It is left for later versions of this document to fill this gap. One way would be to derive a new tcp_ao_secret from the previous tcp_ao_secret and use a new KeyID. However, this could expose the key update event to on-path attackers. Further guidance is required on the severity of this issue and how it could be mitigated.

Later versions of this document will also specify the interactions between this mode of enabling TCP-AO and other TLS mechanisms, such as using pre-shared keys and 0-RTT data, as well as other TCP extensions, such as TCP Fast Open.

A similar extension could be defined for other protocols that derive a security key such as SSH [RFC4253].

5. Security Considerations

TCP-AO provides a protection against the injection of TCP RST. This can impact legitimate connectionless resets, e.g. when an endpoint loses the required state to send TCP-AO segments. Section 7.7 of [RFC5925] provides recommendations to mitigate this effect.

Using TCP-AO with TLS can also inhibit the triggering of the "bad_record_mac" alert that abruptly closes the TLS session when a decryption error occurs. For instance, injected packets will fail the TCP-AO authentication and be ignored by the receiver instead. This also prevents sessionless resets at the TLS level, and similar recommendations to Section 7.7 of [RFC5925] can apply.

6. IANA Considerations

IANA is requested to create a new "Opportunistic TCP-AO with TLS" heading for the new registries defined in this section. New registrations under this heading follow the "Specification Required" policy of [RFC8126].

IANA is requested to add the following entries to the existing "TLS ExtensionType Values" registry.

Value	Extension Name	TLS 1.3	Recommended	Reference
TBD	tcp_ao	CH, EE	N	This document

Table 1

Note that "Recommended" is set to N as this extension is intended for uses as described in this document.

IANA is requested to create a new registry "Authentication Algorithms" under the "Opportunistic TCP-AO with TLS" heading.

The registry governs an 8-bit space. Entries in this registry must include a "Algorithm name" field containing a short mnemonic for the algorithm. Its initial content is presented in Section 4.1 in the TCPAOAuth enum. The registry has a "Reference" column. It is set to [RFC5926] for the two initial algorithms.

IANA is requested to create a new registry "Key Derivation Functions" under the "Opportunistic TCP-AO with TLS" heading.

The registry governs an 8-bit space. Entries in this registry must include a "Key Derivation Function name" field containing a short mnemonic for the function. Its initial content is presented in Section 4.1 in the TCPAOKDF enum. The registry has a "Reference" column. It is set to [RFC5926] for the two initial functions.

Acknowledgments

The authors thank Dimitri Safonov for the TCP-AO implementation in Linux. The authors thank Michael TÃ¼xen, Yoshifumi Nishida and Alessandro Ghedini for their questions and comments on the document during the TCPM meeting at IETF 118.

Change log

References

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>.

[RFC5705] Rescorla, E., "Keying Material Exporters for Transport Layer Security (TLS)", RFC 5705, DOI 10.17487/RFC5705, March 2010, <https://www.rfc-editor.org/rfc/rfc5705>.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, DOI 10.17487/RFC5925, June 2010, <https://www.rfc-editor.org/rfc/rfc5925>.

[RFC5926] Lebovitz, G. and E. Rescorla, "Cryptographic Algorithms for the TCP Authentication Option (TCP-AO)", RFC 5926, DOI 10.17487/RFC5926, June 2010,

<https://www.rfc-editor.org/rfc/rfc5926>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/rfc/rfc8126>.

[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>.

[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>.

Informative References

[CONEXT24] Wirtgen, T., Rybowski, N., Pelsser, C., and O. Bonaventure, "The Multiple Benefits of a Secure Transport for BGP", Association for Computing Machinery (ACM), Proceedings of the ACM on Networking vol. 2, no. CoNEXT4, pp. 1–23, DOI 10.1145/3696406, November 2024, <https://doi.org/10.1145/3696406>.

[RFC4253] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, DOI 10.17487/RFC4253, January 2006, <https://www.rfc-editor.org/rfc/rfc4253>.

Authors' Addresses

Maxime Piraux
UCLouvain
Email: maxime.piraux@uclouvain.be

Olivier Bonaventure
UCLouvain & WELRI
Email: olivier.bonaventure@uclouvain.be

Thomas Wirtgen
UCLouvain
Email: thomas.wirtgen@uclouvain.be