Internet Engineering Task Force (IETF)
Request for Comments: 5925
Obsoletes: 2385
Category: Standards Track
ISSN: 2070-1721

J. Touch
USC/ISI
A. Mankin
Johns Hopkins Univ.
R. Bonica
Juniper Networks

June 2010

The TCP Authentication Option

Опция аутентификации TCP

Аннотация

Этот документ содержит спецификацию опции аутентификации TCP (TCP-AO[1]), которая отменяет опцию TCP MD5 Signature из RFC 2385 (TCP MD5). TCP-AO задает использование более строгих кодов MAC[2], защищающих от повторного использования пакетов даже в долгоживущих соединениях TCP и обеспечивает более четкую привязку ассоциации защиты с соединениями TCP, нежели TCP MD5. Опция TCP-AO совместима со статической настройкой MKT[3] или внешним механизмом установки MKT по отдельному каналу. В обоих случаях TCP-AO защищает соединение при использовании одного MKT для разных экземпляров соединений, используя транспортные ключи, выведенные из MKT, и координирует замену MKT между конечными точками. Это предназначено для поддержки имеющейся инфраструктуры применения TCP MD5, например, защиты долгосрочных соединений (скажем, в BGP и LDP), а также для поддержки большего набора MAC с минимальными изменениями в системах и их работе. TCP-AO использует другой идентификатор опции по сравнению с TCP MD5, даже при том, что TCP-AO и TCP MD5 не могут применяться совместно. Опция TCP-AO поддерживает IPv6 и полностью совместима с предложенными требованиями для замены TCP MD5.

Статус документа

Этот документ содержит проект стандарта Internet (Internet Standards Track).

Документ является результатом работы IETF[4] и представляет собой согласованное мнение членов (сообщества) IETF. Документ был представлен для публичного обсуждения и одобрен для публикации IESG[5]. Дополнительную информацию о стандартах Internet можно найти в разделе 2 документа RFC 5741.

Информацию о текущем состоянии этого документа, обнаруженных ошибках и способах обратной связи можно найти по ссылке http://www.rfc-editor.org/info/rfc5925.

Оглавление

[1] TCP Authentication Option.
[2] Message Authentication Code - код аутентификации (проверки подлинности) сообщения.
[3] Master Key Tuple - кортеж первичного ключа.
[4] Internet Engineering Task Force - комиссия по решению инженерных задач Internet.
[5] Internet Engineering Steering Group - комиссия по инженерным разработкам Internet.

1. Введение

Сигнатура TCP MD5 (TCP MD5) является опцией TCP для проверки подлинности сегментов TCP, включая псевдозаголовок IP, заголовок и данные TCP. Она была предложена для защиты сессий BGP от обманных сегментов TCP, которые могли влиять на данные BGP или отказоустойчивость самих соединений TCP [RFC2385][RFC4953].

По поводу TCP MD5 было высказано много опасений. Использование простой хэш-аутентификации является проблематичным, поскольку были усилены атаки на сам механизм [Wa05]. В TCP MD5 также не достает гибкости управления ключами и алгоритмом. Этот документ добавляет гибкость выбора алгоритма и обеспечивает простой механизм координации ключей, позволяющих менять их в рамках соединения. Однако документ не предусматривает полного криптографического управления ключами по каналу TCP, поскольку сегменты TCP SYN не обеспечивают достаточного для такого согласования пространства (7.6. Влияние на размер заголовка TCP). Документ отменяет использование опции TCP MD5, меняя ее на более общую опцию TCP-AO. Новая опция позволяет применять более строгие функции хэширования, обеспечивает защиту от повторов в долгосрочных соединениях и повторяющихся экземплярах соединений, а также дает более явные рекомендации в части внешнего управления ключами. Результат совместим с IPv6 и полностью совместим с предложенными требованиями по замене TCP MD5 [Ed07].

TCP-AO отменяет TCP MD5, хотя конкретная реализация может для совместимости поддерживать оба механизма. Для данного соединения может применяться лишь один из этих механизмов. Защищенные TCP MD5 соединения нельзя перевести на TCP-AO, поскольку TCP MD5 не поддерживает изменения алгоритма защиты после организации соединения.

1.1. Используемые соглашения

Ключевые слова **необходимо** (MUST), **недопустимо** (MUST NOT), **требуется** (REQUIRED), **нужно** (SHALL), **не нужно** (SHALL NOT), **следует** (SHOULD), **не следует** (SHOULD NOT), **рекомендуется** (RECOMMENDED), **возможно** (MAY), **необязательно** (OPTIONAL) в данном документе должны интерпретироваться в соответствии с RFC 2119 [RFC2119].

В документе эти слова выделяются **полужирным** шрифтом, а обычное использование слов не означает уровня требования в соответствии с RFC 2119.

Символы >> в начале смещенных вправо строк указывают заявления о совместимости с требованиями с использованием указанных выше ключевых слов. Это позволяет быстро найти в документе требования по совместимости.

1.2. Заявление о применимости

Опция TCP-AO предназначена для поддержки имеющегося применения TCP MD5, такого как защита долгосрочных соединений протоколов маршрутизации, например, BGP и LDP. Опция также обеспечивает похожую защиту для долгосрочных соединений TCP и может применяться, например, между кэшами прокси, а не только или главным образом для протоколов маршрутизации.

Опция TCP-AO предназначена для замены (и отмены) применения TCP MD5. TCP-AO расширяет возможности TCP MD5, как отмечено в параграфе 1.3. Краткая сводка рекомендация этого документа приведена ниже.

>> Реализации TCP, поддерживающие TCP MD5, **должны** поддерживать TCP-AO.

>> TCP-AO **следует** реализовать там, где нужна аутентификация TCP по причине отсутствия поддержки IPsec или требуются конкретные свойства TCP-AO (например, ключи на уровне соединения).

>> TCP-AO **можно** реализовать в любом случае.

Опция TCP-AO не предназначена для использования со стеком IPsec (IPsec и IKE[1]) для защиты соединений TCP [RFC4301][RFC4306]. Конкретные различия указаны в параграфе 1.3. Фактически рекомендуется применять IPsec и IKE, особенно там, где желательна поддержка согласования параметров или сеансовых ключей, а также смена ключей. Опция TCP-AO предназначена для применения лишь там, где невозможно реализовать IPsec, например, для

[1]Internet Key Exchange Protocol - протокол обмена ключами в Internet.

поддержки протоколов маршрутизации или когда нужно четко согласовывать ключи с отдельными транспортными сессиями [Ed07].

Опция TCP-AO не предназначена для использования с TLS[1] [RFC5246], Secure BGP (sBGP), Secure Origin BGP (soBGP) [Le09] и иными механизмами, защищающими лишь поток данных TCP. TCP-AO защищает транспортный уровень, предотвращая атаки на сами соединения TCP [RFC4953]. Механизмы защиты потока данных защищают лишь содержимое сегментов TCP и могут быть нарушены воздействием на соединение. Некоторые из указанных протоколов защиты данных (особенно TLS) предлагают более широкий набор механизмов аутентификации и управления ключами, нежели TCP-AO, и таким образом, иначе защищают поток данных. TCP-AO можно применять вместе с защитой потока данных для дополнения.

1.3. Сводка отличий

Этот документ заменяет TCP MD5 [RFC2385] в отмеченных ниже аспектах.

- TCP-AO использует отдельную опцию Kind (29).
- TCP-AO позволяет использовать TCP MD5 для унаследованных соединений.
- TCP-AO заменяет один алгоритм TCP MD5 MAC, алгоритмами MAC, заданными в отдельных документах, и может быть расширен включением других MAC.
- TCP-AO позволяет менять ключи в соединении TCP при условии смены ключей по отдельному протоколу (out-of-band) или вручную. Опция включает идентификатор ключа (key ID), который эффективно позволяет использовать одновременно несколько ключей и механизм координации ключей «получить идентификатор следующего ключа» для смены ключей. Отметим, что TCP MD5 не препятствует смене ключей в соединении, но не требует поддержки этого. Кроме того, TCP-AO поддерживает смену ключей без потери сегментов, тогда как в TCP MD5 при смене ключей сегменты могут теряться во время перехода или может требоваться использования для полученных в процессе перехода сегментов разных ключей, поскольку ключи не имеют идентификаторов. Хотя TCP восстанавливает потерянные при смене ключа сегменты, это может существенно влиять на производительность.
- TCP-AO обеспечивает автоматическую защиту от повторов (replay) для долгосрочных соединений с помощью порядковых номеров.
- TCP-AO обеспечивает ключи трафика для соединения, уникальные для каждого соединения TCP, с использованием начальных порядковых номеров TCP (ISN[2]) для различения ключей даже при статических MKT в повторяющихся экземплярах соединений через одну пару сокетов.
- TCP-AO задает детали взаимодействия опции с состояниями TCP, обработкой событий и интерфейсом пользователя.
- TCP-AO на 2 байта короче опции TCP MD5 (16 вместо 18) в заданном изначально варианте (96 битов MAC).

Отличия TCP-AO от решения IPsec/IKE [RFC4301][RFC4306] перечислены ниже.

- TCP-AO не поддерживает динамического согласования параметров.
- TCP-AO включает пару сокетов TCP (адреса и порты отправителя и получателя) в качестве индекса параметров защиты (вместе с KeyID), не используя отдельного поля, как IPsec Security Parameter Index (SPI).
- TCP-AO вынуждает менять рассчитанные MAC при перезапуске соединения даже при сохранении пары сокетов TCP (адреса и номера портов) [Ed07].
- TCP-AO не поддерживает шифрования.
- TCP-AO не аутентифицирует сообщения ICMP (при использовании IPsec некоторые сообщения ICMP аутентифицируются в зависимости от конфигурации).

2. Опция аутентификации TCP

Опция TCP-AO использует значение Kind = 29. В последующих параграфах описана опция TCP-AO и приведен обзор TCP MD5 для сравнения.

2.1. Обзор опции TCP MD5

Структура опции TCP MD5 представлена на рисунке 1.

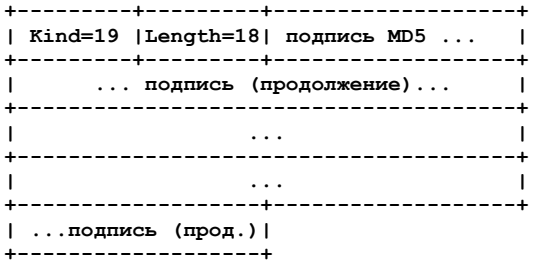

Рисунок 1. Опция TCP MD5 [RFC2385].

В опции TCP MD5 поле размера фиксировано и дайджест (подпись) MD5 занимает 16 байтов вслед за полями Kind и Length (каждое по 1 байту), а полный размер подписи MD5 составляет 128 битов [RFC1321].

[1]Transport Layer Security - защита транспортного уровня.
[2]Initial Sequence Number.

Опция TCP MD5 задает использование подписи MD5 для указанных ниже полей с соблюдением порядка.

1. Псевдозаголовок IP (IP-адреса отправителя и получателя, номер протокола, размер сегмента).
2. Заголовок TCP без опций и контрольной суммы.
3. Данные TCP (payload).
4. Ключ

2.2. Формат опции TCP-AO

TCP-AO обеспечивает надмножество возможностей TCP MD5 и минимальная в духе SP4 [SDNS88]. Опция TCP-AO включает поля Kind и поле Length как TCP MD5, а также поля KeyID и RnextKeyID, показанные на рисунке 2.

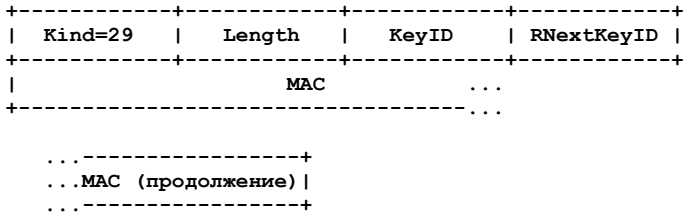

Рисунок 2. Опция TCP-AO.

Определения полей TCP-AO приведены ниже.

Kind

1-байтовое целое число без знака, указывающее опцию TCP-AO. Kind = 29.

>> Конечной точке **недопустимо** использовать TCP-AO в одном соединении с опцией TCP MD5. При наличии обеих опций протокол TCP **должен** отбрасывать сегмент без уведомления отправителя.

>> Одному сегменту TCP **недопустимо** включать долее 1 опции TCP-AO. При наличии нескольких опций протокол TCP **должен** отбрасывать сегмент без уведомления отправителя.

Length

1-байтовое целое число без знака, указывающее размер опции в байтах, включая поля Kind, Length, KeyID, RnextKeyID и MAC.

>> Поле Length **должно** иметь значение не меньше 4, в противном случае TCP **должен** отбрасывать сегмент без уведомления отправителя.

>> Значение поля Length должно соответствовать размеру заголовка TCP, в противном случае TCP **должен** отбрасывать сегмент без уведомления отправителя.

Проверка поля Length предполагает, что сумма размеров всех опций при сложении с размером базового заголовка TCP (5 слов) совпадает со значением поля TCP Offset. Такая полная проверка возможна, поскольку RFC 793 задает размер требуемых опций, а RFC 1122 требует, чтобы все новые опции использовали единый формат с фиксированным размером [RFC793][RFC1122]. Проверка может быть частичной с учетом лишь опции TCP-AO, когда размер TCP-AO в сумме со смещением TCP-AO от начала заголовка TCP не первышает размер заголовка TCP, указанный в поле Offset заголовка TCP.

Значение 4 и другие небольшие значения > 4 (например, указывающие очень короткие поля MAC) не запрещены, но бессмысленны для этой опции.

KeyID

1-байтовое целое число без знака, указывающее кортеж первичного ключа (MKT, параграф 3.1), служащий для генерации ключей трафика, который применяются при создании MAC для аутентификации данного сегмента.

Поддерживается эффективная смена ключей в процессе работы соединения и/или помощь в координации ключей при организации соединения, как описано в параграфе 6.1. Отметим, что KeyID не имеет криптографических свойств и значение должно ьыть случайным без какого-либо резервирования.

>> KeyID **может** совпадать для обоих направления соединения, но это не требуется и не имеет особого смысла.

Это позволяет устанавливать MKT в группе устройств без предшествующей координации KeyID среди них, а также позволяет добавлять новые устройства в группу MKT без перенумерации KeyID. Эти два свойства особенно важны при использовании с шаблонами пар сокетов TCP для MKT, т. е. при использовании MKT для группы устройств, заданных шаблоном (3.1. Кортеж первичного ключа (MKT)).

RnextKeyID

1-байтовое целое число без знака, указывающее MKT, готовый у отправителя для аутентификации принятого сегмента, т. е. желаемый идентификатор ключа для следующего принимаемого сегмента.

Обеспечивается эффективная координация смены ключей (6.1. Координация применения новых MKT). Поле RNextKeyID не имеет криптографических свойств и нужно применять случайное, а не зарезервированное значение.

MAC (Message Authentication Code)

Содержимое поля определяется параметрами защитной ассоциации. Типичные MAC имеют размер 96-128 битов (12-16 байтов), но разрешен любой размер, помещающийся в аутентифицируемый сегмент. Расчет MAC описан в параграфе 5.1. Алгоритмы MAC.

>> Требуется поддержка TCP-AO MAC, определенных в [RFC5926] и **могут** поддерживаться другие MAC.

Поля TCP-AO не указывают алгоритм MAC явно (как TCP MD5) или неявно. Конкретный алгоритм является частью конфигурации защитной ассоциации и устанавливается отдельно (3. Ключи TCP-AO и их свойства).

Следует отметить, что TCP-AO не влияет на анонсируемый максимальный размер сегмента TCP (Maximum Segment Size или MSS), как другие опции TCP [Bo09].

В оставшейся части документа описана обработка TCP-AO и связь опции с TCP.

3. Ключи TCP-AO и их свойства

Работа TCP-AO основана на двух наборах ключей для аутентификации входящих и исходящих сегментов - MKT, а также ключах трафика. MKT служат для создания уникальных ключей трафика и содержат ключевой материал для генерации этих ключей, а также указывают параметры, с которыми применяются ключи. К таким параметрам относится

набор аутентифицируемых опций TCP, а также индикаторы алгоритмов, применяемых для вывода ключей трафика и расчета MAC. Ключи трафика являются материалом при расчете MAC для отдельных сегментов TCP.

3.1. Кортеж первичного ключа (MKT)

Кортеж первичного ключа (MKT) описывает свойства опции TCP-AO, связанной с 1 или несколькими соединениями.

TCP connection identifier - идентификатор соединения TCP

Пара сокетов TCP (локальные и удаленные адреса IP и номера портов TCP). Значения можно задавать частично с помощью диапазона (например, 2-30), маски (например, 0xF0), шаблона (например, "*") или иным способом.

TCP option flag - флаг опции TCP

Этот флаг указывает, включаются ли отличные от TCP-AO опции TCP в расчет MAC. При включении опций их содержимое в порядке указания включается в MAC с использованием значения 0 для поля TCP-AO MAC. Если опции не включены, в расчете учитывается лишь TCP-AO, а прочие опции пропускаются (без обнуления). Отметим, что опция TCP-AO учитывается всегда (с обнулением поля MAC), независимо от установки данного флага. Это защищает индикацию размера MAC, а также поля идентификаторов ключей (KeyID, RNextKeyID). Флаг применим к общим TCP в обоих направлениях (входящие и исходящие сегменты).

IDs - идентификаторы

Значения, используемые в полях KeyID и RNextKeyID опции TCP-AO и служащие для различения MKT при одновременном использовании (KeyID), а также для индикации MKT, готовых для использования в противоположном направлении (RnextKeyID).

Каждый MKT имеет два идентификатора - SendID и RecvID. Первый помещается в поле KeyID опции TCP-AO в исходящих сегментах, а RecvID сопоставляется с TCP-AO KeyID во входящих сегментах. Применение идентификаторов рассмотрено в параграфах 7.4 и 7.5.

>> MKT ID **должны** поддерживать любые значения от 0 до 255, включительно. Резервных значений нет.

Значения ID выделяются произвольно, т. е. не требуется их монотонного роста, нет резервных значений и им не придается какого-либо особого смысла. Значения могут выделяться по порядку или на основе любого метода, согласованного конечными точками соединения (например, с помощью механизма внешнего управления MKT).

>> **Недопустимо** предполагать случайное назначение ID.

Master key - первичный ключ

Последовательность байтов, применяемая для генерации ключей трафика, которая может быть выведена из общего ключа, распространяемого внешним протоколом по отдельному каналу. Последовательность используется алгоритмом создания ключей трафика, описанным в параграфе 5.2.

Реализациям предлагается хранить значения первичного ключа в защищенной приватной области памяти или иного хранилища.

Key Derivation Function (KDF) - функция вывода ключей

Указывает функцию вывода ключей и их параметров, используемую для генерации ключей трафика из первичных ключей. Описание приведено в параграфе 5.2 и в [RFC5926].

Message Authentication Code (MAC) algorithm - алгоритм кода аутентификации сообщений

Указывает алгоритм MAC и его параметры, применяемые для данного соединения. Дополнительная информация приведена в параграфе 5.1 и [RFC5926].

>> Компоненты MKT **недопустимо** менять в течение работы соединения.

Невозможность смены значений MKT в соединении обусловлена тем, что состояние TCP координируется в процессе организации соединения. В TCP нет процедур для смены этого состояния, когда соединение уже есть.

>> Набор MKT **можно** менять в процессе работы соединения.

Параметры MKT не меняются и можно установить новые MKT и соединение может сменить применяемый MKT.

>> **Недопустимо** перекрытие идентификаторов MKT в случае перекрытия идентификаторов соединений TCP.

Этот документ не задает способ создания MKT пользователем или процессом. Предполагается, что MKT, влияющий на конкретное соединение, не может быть уничтожен для активного соединения или (что эквивалентно) его параметры копируются в локальную область соединения (создается экземпляр) и поэтому изменения будут влиять лишь на новые соединения. Для управления MKT может применяться отдельный прикладной протокол.

3.2. Ключи трафика

Ключ трафика выводится и MKT и пар адресов IP и номеров портов TCP (локальные и удаленные), а для организованных соединений - исходных порядковых номеров TCP (Initial Sequence Number или ISN) в каждом направлении. Сегменты, переданные до организации соединения используют ту же информацию, заменяя неизвестные значения 0 (например, несогласованные ISN).

Один кортеж MKT можно использовать для создания 4 разных ключей трафика:

- Send_SYN_traffic_key;
- Receive_SYN_traffic_key;
- Send_other_traffic_key;
- Receive_other_traffic_key.

Отметим, что ключи являются односторонними. Данной соединение обычно применяет лишь 3 из этих ключей, поскольку обычно используется только один из ключей SYN. Все 4 ключа применяются при «одновременном открытии» соединения [RFC793].

Связи между MKT и ключами трафика показаны на рисунке 3, где ключи трафика помечены символом *. Отметим, что каждый MKT можно использовать для создания любого из 4 ключей трафика, но рассчитывать требуется лишь ключи, реально нужные для обработки сегментов соединения. Подробное описание генерации ключей дано в параграфе 5.2.

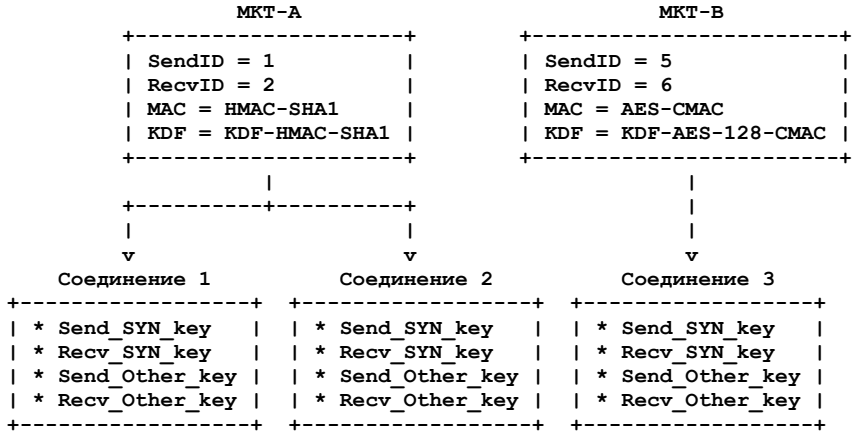

Рисунок 3. Связь между MKT и ключами трафика.

3.3. Свойства MKT

TCP-AO требует, чтобы каждый защищаемый сегмент TCP соответствовал в точности одному MKT. Когда исходящий сегмент соответствует MKT, применяется опция TCP-AO, в противном случае она не используется. Одному исходящему сегменту может соответствовать несколько MKT, например при смене MKT. Эти MKT не могут иметь конфликтующих идентификаторов, и должен быть некий механизм, определяющий MKT, применяемый для данного исходящего сегмента.

>> Исходящий сегмент TCP **должен** соответствовать не более, чем одному желаемому MKT, указанному парой сокетов сегмента. Сегмент **может** соответствовать нескольким MKT, при условии указания единственного желаемого MKT. Для определения желаемого MKT **может** использоваться другая информация в сегменте, но в нее **недопустимо** включать значения полей каких-либо опций TCP.

Рекомендуется использовать в механизме выбора MKT лишь ту информацию, которую TCP-AO будет аутентифицировать. Поскольку MKT могут указывать, что опции, не относящиеся к TCP-AO, можно игнорировать при расчете MAC, рекомендуется не применять эти опции при выборе MKT.

>> Входящий сегмент TCP, включая TCP-AO, **должен** соответствовать в точности одному MKT, указанному исключительно парой сокетов и TCP-AO KeyID.

Входящие сегменты включают внутри TCP-AO индикатор выбора среди подходящих MKT - поле KeyID. TCP-AO требует, чтобы для выбора среди подходящих MKT применялось лишь поле KeyID, что позволяет координировать смену MKT с помощью смены ключа в TCP-AO.

>> Когда исходящий сегмент TCP не соответствует никакому MKT, опция TCP-AO не применяется. TCP-AO всегда используется для исходящих сегментов, соответствующих MKT, и не применяется в иных случаях.

4. Параметры TCP-AO для соединения

TCP-AO использует небольшое число параметров, связанных с каждым соединение, которое использует TCP-AO после организации. Эти значения могут сохраняться в блоке управления транспортом (Transport Control Block или TCB) [RFC793]. Значения описаны в последующих параграфах документа, а список их приведен ниже.

1. *Current_key* - MKT, применяемый в настоящее время для аутентификации исходящих сегментов, где SendID помещается в сегмент как KeyID (параграф 7.4, 2.f). Входящие сегменты аутентифицируются с помощью MKT, соответствующего сегменту и его TCP-AO KeyID (параграф 7.5, 2.c) по сопоставлению с MKT идентификатора соединения TCP и MKT RecvID. В каждый момент для данного соединения имеется лишь одно значение current_key.

   >> Каждое соединение TCP в состоянии, отличном от IDLE, **должно** иметь не более одного current_key.

2. Rnext_key - MKT, предпочитаемый в настоящее время для входящих (принимаемых) сегментов, где RecvID помещен в исходящий сегмент как RNextKeyID (параграф 7.4, 2.d).

   >> Каждое соединение TCP в состоянии, отличном от IDLE, **должно** иметь не более одного rnext_key.

3. Пара SNE (Sequence Number Extension - расширение с порядковым номеров). SNE служат для предотвращения атак с повторным использованием (replay), как описано в параграфе 6.2. Каждый SNE инициализируется значением 0 при создании соединения. Использование номеров при расчете MAC описано в параграфе 5.1.

4. Один или несколько кортежей MKT, которые соответствуют паре сокетов данного соединения.

   MKT применяются вместе с другими параметрами соединения для создания уникальных ключей трафика в каждом соединении, как описано в параграфе 5.2. Эти ключи могут кэшироваться после расчета и сохраняться в TCB с соответствующей информацией MKT. Они могут считаться частью параметров соединения.

5. Криптографические алгоритмы

TCP-AO использует криптографические алгоритмы для расчета MAC (код аутентификации сообщения), применяемого при проверке подлинности сегмента и его заголовков, которые называют алгоритмами MAC. Эти алгоритмы задаются в отдельных документах для упрощения независимого от протокола обновления требований к алгоритмам [RFC5926]. TCP-AO также применяет криптографические алгоритмы для преобразования MKT, которые могут совместно использоваться несколькими соединениями, в уникальные для каждого соединения ключи трафика. Это называется

функциями вывода ключей (Key Derivation Function или KDF) и также задано в [RFC5926]. В данном разделе описано использование алгоритмов в TCP-AO.

5.1. Алгоритмы MAC

Алгоритмы MAC принимают на входе данные переменного размера и ключ, возвращая на выход число фиксированного размера. Это число служит для определения, поступили ли входные данные от источника с тем же ключом и не были ли они подменены в процессе передачи. Интерфейс кодов MAC для TCP-AO показан ниже.

```
MAC = MAC_alg(traffic_key, message)
```

Ввод: MAC_alg, traffic_key, message

Вывод: MAC

Определения элементов интерфейса представлены ниже.

MAC_alg
Конкретный алгоритм MAC, применяемый для расчета. Алгоритм MAC задает выходной размер, поэтому отдельного поля размера не требуется, см. [RFC5926].

Traffic_key
Ключ трафика, используемый для расчета. Расчет выполняется с текущим MKT соединения (параграф 5.2).

Message
Входные данные, для которых рассчитывается MAC. В TCP-AO это сегмент TCP, которому предшествует псевдозаголовок IP и заголовок TCP, как описано в этом параграфе.

MAC
Выход алгоритма MAC с заданными параметрами (постоянный размер).

На момент написания этого документа, для алгоритмов, применяемых в TCP-AO, как описано в [RFC5926], задана отсечка до 96 битов. Хотя каждый из алгоритмов дает MAC большего размера, 96 битов обеспечивают разумный компромисс между защищенностью и размером сообщения. Однако в будущем это может измениться, поэтому размер TCP-AO не следует предполагать постоянным.

Алгоритм MAC, используемый для расчета кода MAC в соединении, выполняется в соответствии с определением MKT из [RFC5926].

Обязательные для реализации алгоритмы MAC в опции TCP-AO описаны в отдельном RFC [RFC5926]. Это позволяет выполнить для спецификации TCP-AO процесс стандартизации IETF Standards Track, даже при возникновении потребности в замене алгоритмов и их меток (которые могут применяться на пользовательском интерфейсе и в протоколах автоматизированного управления MKT) в результате изменений в динамичном мире криптографии.

>> **Могут** поддерживаться дополнительные алгоритмы, не требуемые в TCP-AO.

Входом данных для MAC служат указанные ниже поля (с сохранением порядка) с сетевым порядком байтов.

1. Порядковый номер (SNE) с сетевым порядком байтов, как показано ниже (см. также параграф 6.2).

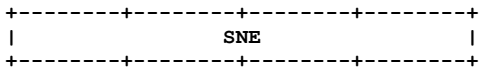

Рисунок 4. Порядковый номер.

SNE для передаваемых сегментов поддерживаются локально в переменной SND.SNE, а для принятых сегментов используется локальное значение RCV.SNE. Детали поддержки и использования этих значений приведены в параграфах 6.2, 7.4 и 7.5.

2. Псевдозаголовок IP с адресами IP для отправителя и получателя, номером протокола и размером сегмента, заданными в сетевом порядке байтов, предшествующий заголовкуTCP. Псевдо заголовок IP не отличается от применяемого для расчета контрольной суммы TCP в случаях IPv4 [RFC793] и IPv6 [RFC2460].

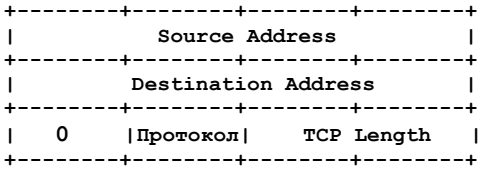

Рисунок 5. Псевдозаголовок IPv4 для TCP.

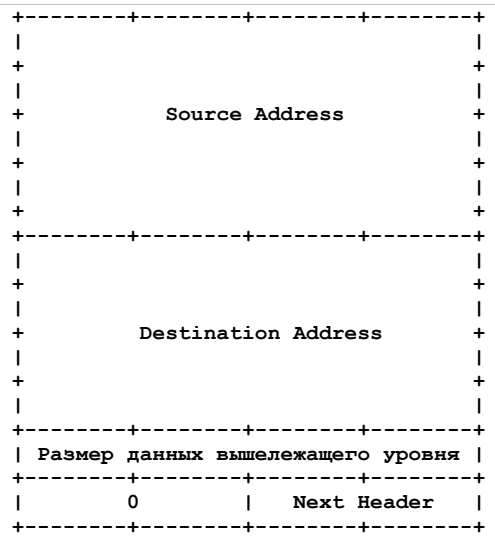

Рисунок 6. Псевдозаголовок IPv6 [RFC2460].

3. Заголовок TCP, включающий по умолчанию опции, где при расчете поля контрольной суммы TCP и TCP-AO MAC принимаются нулевыми. Байты располагаются в сетевом порядке.

   Флаг опций TCP в MKT определяет включение полей опций TCP в расчет MAC. При включении опций для расчета обнуляется лишь поле TCP-AO MAC. Если флаг указывает исключение опций, все опции TCP, за исключением TCP-AO, опускаются при расчете MAC. Значение поля TCP-AO MAC при расчете принимается нулевым.

4. Данные TCP, т. е. содержимое сегмента TCP.

   Отметим, что ключи трафика не считаются частью данных, их использование задает алгоритм MAC (например, в качестве HMAC [RFC2104][RFC2403]). Ключи трафика выводятся из текущего MKT, как указано в параграфе 5.2.

5.2. Функции выведения ключей трафика

Ключи трафика выводятся из текущих MKT с использованием функций KDF, как показано ниже.

```
traffic_key = KDF_alg(master_key, context, output_length)
```

Ввод: KDF_alg, master_key, context, output_length

Вывод: traffic_key

Описание параметров создания ключей трафика представлено ниже.

KDF_alg
Конкретная функция KDF, служащая основным элементом создания ключей трафика в соответствии с текущим MKT (см. [RFC5926]).

Master_key
Строка первичного ключа (master_key), хранящаяся в связанном MKT.

Context
Контекст, служащий вводом при создании ключа трафика traffic_key, как указано в [RFC5926]. Конкретный способ использования контекста и других данных для обеспечения необработанного вывода KDF описан в [RFC5926].

Output_length
Желаемый размер вывода KDF, т. е. размер, до которого этот вывод отсекается в соответствии с [RFC5926].

Traffic_key
Желаемый вывод KDF размером output_length, для использования в качестве входных данных алгоритма MAC, как описано в параграфе 5.1.

Контекст, служащих вводом для KDF, включает пару сокетов TCP и начальные порядковые номера (ISN) для соединения. Эти данные уникальны для каждого экземпляра соединения TCP, что позволяет TCP-AO создавать уникальные ключи трафика для соединения даже при использовании MKT на нескольких соединениях или при повторяющихся соединениях с той же парой сокетов. Контекст KDF показан на рисунках 7 и 8.

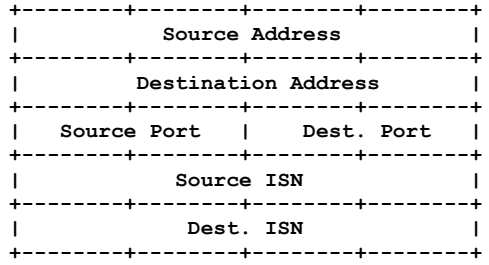

Рисунок 7. Контекст KDF для соединения IPv4.

Ключи трафика являются «направленными», поэтому отправитель и получатель интерпретируются по-разному для входящих и исходящих сегментов. Во входящих сегментах отправителем является удаленный сайт, в исходящих - локальный. Это гарантирует уникальность ключей для каждого направления.

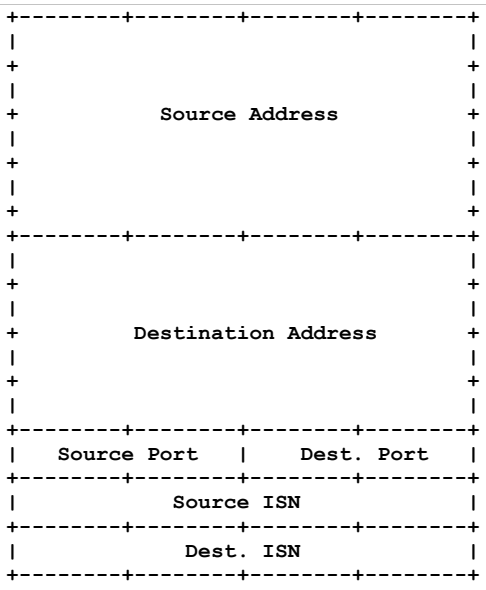

Рисунок 8. Контекст KDF для соединения IPv6.

Для сегментов с флагом SYN (но без ACK) значение ISN получателя неизвестно. Поэтому для них ключ соединения рассчитывается с использованием показанного выше контекста с нулевым значением Destination ISN. Для прочих сегментов используется известная пара ISN. Если значение ISN неизвестны (например, при отправке RST после перезагрузки), сегмент следует передавать без аутентификации. Если же аутентификация требуется, создать корректный код MAC все равно не удастся и сегмент будет отброшен при получении.

>> Сегменты TCP-AO SYN (SYN, без ACK) **должны** использовать Destination ISN = 0 (отправляемые и принимаемые), все прочие сегменты используют известную пару ISN.

В целом это означает, что каждое соединение будет иметь 4 разных ключа трафика для каждого MKT, как показано ниже.

Send_SYN_traffic_key
Ключ трафика, служащий для аутентификации исходящих сегментов SYN. Значение Source ISN известно (локальный ISN соединения TCP), Destination ISN (удаленный) неизвестно (используется 0).

Receive_SYN_traffic_key
Ключ трафика, служащий для аутентификации входящих сегментов SYN. Значение Source ISN известно (удаленный ISN соединения TCP), Destination ISN неизвестно (используется 0).

Send_other_traffic_key
Ключ трафика, служащих для аутентификации всех прочих исходящих сегментов TCP.

Receive_other_traffic_key
Ключ трафика, служащих для аутентификации всех прочих входящих сегментов TCP.

В приведенной ниже таблице показано, как рассчитывается каждый из этих ключей. Алгоритмы TCP-AO используют адреса IP (IP) для отправителя (S) и получателя (D), порты TCP (port), номера ISN (ISN) и каждый сегмент (входящий или исходящий) помечен как относящийся к локальной (l) или удаленной ® стороне.

	S-IP	***S-port***	***S-ISN***	***D-IP***	***D-port***	***D-ISN***
Send_SYN_traffic_key	l-IP	l-port	l-ISN	r-IP	r-port	0
Receive_SYN_traffic_key	r-IP	r-port	r-ISN	l-IP	l-port	0
Send_other_traffic_key	l-IP	l-port	l-ISN	r-IP	r-port	r-ISN
Receive_other_traffic_key	r-IP	r-port	r-ISN	l-IP	l-port	l-ISN

Использование обоих номеров ISN в расчетах ключей трафика гарантирует предотвращение повторного использования сегментов в повторяющихся соединениях через те же сокеты. 32-битовые номера позволяют избавиться от повторов за исключением случаев перезагрузки. Повторное использование предполагает, что обе стороны повторяют свое использование в том же соединении. Ожидается, что:

>> конечным точкам **следует** выбирать номера ISN псевдослучайно, например, как указано в [RFC1948].

Сегменты SYN аутентифицируются с нулевым значением ISN получателя (при отправке и приеме), а все прочие сегменты используют пару ISN для соединения. Имеются другие случаи, когда значение Destination ISN неизвестно, но сегменты передаются (например после перезагрузки конечной точки), хотя у конечных точек может не быть информации, достаточной для аутентификации сегментов. Это дополнительно рассматривается в параграфе 7.7.

5.3. Вопросы организации и срока действия ключей трафика

TCP-AO не предоставляет механизма согласования ключей трафика или параметров (алгоритм MAC, размер, использование TCP-AO для соединения), а также смены ключей в соединении. Предполагается использование механизмов распространения MKT, согласования параметров и смены ключей по отдельным каналам. Разделение использования и поддержки MKT похоже на применяемое в стеке IPsec [RFC4301][RFC4306].

Пользователям TCP-AO предлагается применять известные методы генерации подходящих MKT, включая обоснованный размер первичного ключа, ограничение совместного использования ключей трафика и продолжительности действия MKT в соответствии с [RFC3562]. Этот также включает использование в соединении отдноразовых значений (nonce), как предложено в параграфе 5.2[1].

[1]В параграфе 5.2 об этом ничего не сказано. *Прим. перев.*

TCP-AO поддерживает смену ключей с согласованием и координацией новых MKT по отдельному каналу с помощью протокола или вручную [RFC4808]. Новые MKT применяются скоординировано с использованием отдельного (out-of-band) механизма для обновления обеих конечных точек TCP. Когда в каждый момент применяется лишь один MKT, временное использование недействительного MKT может приводить к отбрасыванию сегментов. Хотя протокол TCP достаточно устойчив к такому отбрасыванию, в TCP-AO применяется поле KeyID, позволяющее избежать потерь. Данное соединение может иметь несколько соответствующих MKT, а поле KeyID служит для выбора MKT, соответствующего используемому для сегмента ключу трафика для предотвращения необходимости выбора MKT методом проб и ошибок.

TCP-AO предоставляет явный механизм координации MKT, описанный в параграфе 6.1. такой механизм полезен при установке новых MKT или смене MKT для определения момента начала использования установленных MKT.

Пользователям рекомендуется поддерживать MKT в соответствии с рекомендациями по управления ключами при использовании TCP MD5 [RFC3562], в частности, применять подходящий размер ключей (12-24 байта) и избегать совместного использования MKT во множестве партнерских связей BGP.

5.3.1. Повторное использование MKT для пары сокетов

MKT можно неоднократно применять для разных пар сокетов в рамках хоста или в разных экземплярах соединения для той же пары сокетов на хосте. В любом случае обеспечивается защита от повторного использования сегментов (replay).

Повторное использование MKT для разных пар сокетов не открывает возможность для replay-атак, поскольку пара сокетов TCP включается в MAC и генерацию ключей трафика. Повторное использование в повторяющихся экземплярах соединений с той же парой сокетов также не способствует таким атакам, поскольку в ключи трафика включаются ISN соединения, а повтор пары ISN маловероятен в разумные сроки.

5.3.2. Использование MKT в долгосрочных соединениях

TCP-AO использует расширение для порядковых номеров (SNE) с целью предотвращения атак на долгосрочные соединения. Явное обновление MKT с помощью внешних механизмов и индексирование на основе поля KeyID можно использовать для смены ключевого материала по разным причинам (например, при смене персонала), но это не требуется для поддержки долгосрочных соединений.

6. Дополнительные механизмы защиты

TCP-AO добавляет механизмы поддержки активного использования, особенно в средах, где доступна поддержка ключей лишь вручную. Это включает описанные ранее механизмы поддержки множества одновременных MKT (с помощью поля KeyID) и генерацию уникальных ключей трафика для соединения (с помощью KDF). В этом разделе описаны дополнительные механизмы координации смены MKT и предотвращения replay-атак при продолжительном использовании ключа трафика.

6.1. Координация применения новых MKT

В любой конкретный момент соединение TCP может иметь несколько MKT, заданных для сегментов каждого направления (входящие, исходящие). TCP-AO обеспечивает механизм индикации готовности нового MKT, что позволяет отправителю начать использование этого нового MKT. Этот механизм позволяет скоординировать применение новых MKT для предотвращения неоправданных потерь при аутентификации отправителя с использованием MKT, еще не готового у получателя.

Отметим, что это предназначено для оптимизации. Решение о начале применения нового ключа является вопросом производительности. Предполагается удаление недействительных MKT. TCP-AO не предоставляет механизмов для координации такого удаления, поскольку это считается операцией управления ключами.

Использование нового MKT координируется с помощью двух идентификаторов в заголовке:

- KeyID;
- RnextKeyID.

KeyID представляет информацию исходящего MKT, применяемую отправителем сегмента для создания MAC (исходящий), а соответствующая информация о входных ключах применяется получателем для проверки этого MAC и включает SendID для MKT, применяемого сейчас в данном направлении.

RNextKeyID представляет сведения о предпочтительном MKT для следующих принимаемых сегментов (receive next), что обеспечивает отправителю сегмента способ указать готовность входящего MKT для принимаемых впредь сегментов и позволяет получателю сегмента узнать, когда следует сменить MKT (а также KeyID и связанные с ними ключи трафика). Это RecvID первичного ключа MKT, желаемого во входящих сегментах.

Имеется 2 указателя, сохраняемые на каждой стороне соединения и содержащие сведения для соединения (раздел 4):

- активный в данный момент исходящий ключ MKT (current_key);
- текущее предпочтение для кходящего MKT (rnext_key).

Current_key указывает MKT, используемый для аутентификации исходящих сегментов, а при организации соединения - первый ключ MKT, выбранный для использования.

Rnext_key указывает входящий MKT, который готов и является предпочтительным для использования, а при организации соединения - активный в данный момент входящий MKT. Значение может быть изменено при установке новых MKT (например, протоколом автоматической смены MKT или ручным выбором пользователя).

Rnext_key меняется лишь вручную пользователем или операцией протокола управления MKT, но не TCP-AO. Current_key обновляется TCP-AO при обработке принятых сегментов TCP, как описано в параграфе 7.5. Отметим, что алгоритм позволяет сменить current_key на новый MKT, а затем вернуться к прежнему MKT («резервное

копирование»). Это может происходить при смене MKT, когда сегменты принимаются с нарушением порядка, и считается свойством TCP-AO, поскольку нарушение порядка не ведет к отбрасыванию сегмента. Единственным способом предотвратить повторное использование прежнего MKT является удаление MKT, признанных ненужными.

6.2. Предотвращение повторов в долгосрочных соединениях

TCP использует 32-битовый порядковый номер, который может повторяться в долгосрочных соединениях. Это может приводить к преднамеренному и легитимному повторному использованию сегментов TCP в соединениях. TCP-AO предотвращает replay-атаки и для этого нужно отличать допустимые повторы от прочих, для чего используется расширение порядковых номеров в форме 32-битовых SNE в принимаемых и передаваемых сегментах.

SNE расширяют порядковые номера TCP так, что сегменты в одном соединении всегда уникальны. При переходе порядкового номера TCP от максимума к началу, возникает вероятность полного повтора сегмента. Использование SNE позволяет различать идентичные сегменты с одинаковыми номерами в разные моменты работы соединения. TCP-AO имитирует 64-битовое пространство порядковых номеров, определяя рост старших 32-битов номера (SNE) по переходу младшей части (TCP) через максимум.

TCP-AO поддерживает SND.SNE для передаваемых сегментов и RCV.SNE для принимаемых, инициализируя их значением 0 при организации соединения. Назначение SNE состоит в организации вместе с 32-битовыми порядковыми номерами TCP единого 64-битового пространства номеров.

Для передаваемых сегментов SND.SNE можно реализовать путем расширения имеющегося порядкового номера TCP до 64 битов, при этом SND.SNE содержит старшие 32 бита номера. Для принимаемых сегментов TCP-AO приходится имитировать использование 64-битовых номеров и корректно выводить 32 старших бита номера RCV.SNE из полученного 32-битового порядкового номера и текущего контекста соединения.

Реализация SNE не задается в этом документе, но одним из возможных способов является использование RCV.SNE, SND.SNE или обоих номеров.

Рассмотрим реализацию с двумя SNE (SND.SNE, RCV.SNE) и приведенными ниже дополнительными переменными, которые вместе с текущим номером сегмента TCP (SEG.SEQ) инициализируются нулями:

- SND.PREV_SEQ для детектирования перехода SND.SEQ через максимум;
- RCV.PREV_SEQ для детектирования перехода RCV.SEQ через максимум;
- SND.SNE_FLAG для индикации инкремента SND.SNE;
- RCV.SNE_FLAG для индикации инкремента RCV.SNE.

При получении сегмента приведенный ниже код рассчитывает SNE, использованный в MAC, для стороны RCV (на стороне SND можно применить эквивалентный алгоритм).

```
/* Установка флага при первом переходе SEG.SEQ через максимум */
if ((RCV.SNE_FLAG == 0)
   && (RCV.PREV_SEQ > 0x7fffffff) && (SEG.SEQ < 0x7fffffff[1])) {
      RCV.SNE = RCV.SNE + 1;
      RCV.SNE_FLAG = 1;
}
/* Выбор SNE для использования после инкрементирования */
if ((RCV.SNE_FLAG == 1) && (SEG.SEQ > 0x7fff)) {
   SNE = RCV.SNE - 1; # use the pre-increment value
} else {
   SNE = RCV.SNE; # use the current value
}
/* Сброс флага в «середине» окна */
if ((RCV.PREV_SEQ < 0x7fffffff) && (SEG.SEQ > 0x7fffffff)) {
   RCV.SNE_FLAG = 0;
}
/* Сохранение текущего SEQ для следующего применения кода */
RCV.PREV_SEQ = SEG.SEQ;
```

При первом переходе порядкового номера через максимум, т. е. когда новый номер (из нижней половины) меньше меньше старого (из верхней половины), устанавливается флаг и инкрементируется SNE.

Если флаг установлен и встречается большой номер, требуется восстановление порядка сегментом, поэтому используется прежнее (до инкрементирования) значение SNE.

К моменту достижения максимум флаг будет сброшен.

Флаг предотвращает повторное инкрементирование SNE и сбрасывается в середине окна (когда старый номер из нижней половины, а новый из верхней). Поскольку размер окна приема никогда не превышает половину пространства номеров, невозможно установить и сбросить флаг одновременно - остающиеся в сети сегменты независимо от нарушения порядка не могут быть одновременно в обеих половинах пространства номеров.

7. Взаимодействие TCP-AO с TCP

Ниже описано влияние TCP-AO на состояния, сегменты, события и интерфейсы TCP. Описание предназначено для дополнения сведений о TCP, как это предусмотрено в [RFC793].

7.1. Пользовательский интерфейс TCP

Пользовательский интерфейс TCP активно и пассивно поддерживает команды OPEN, SEND, RECEIVE, CLOSE, STATUS, ABORT. TCP-AO не меняет этот интерфейс применительно к TCP, но некоторые команды или последовательности команд нужно изменить для поддержки TCP-AO. Детали этих изменений в TCP-AO не задаются.

[1] Здесь и ниже в псевдокоде ошибочно указано 0x7fff вместо 0x7fffffff (https://www.rfc-editor.org/errata/eid5672). *Прим. перев.*

TCP-AO требует расширения пользовательского интерфейса TCP для обеспечения возможности настройки MKT, а также поддержки входящих вызовов для задания активного MKT. Ключи MKT должны быть настроены до организации соединения и набор MKT может изменяться во время соединения.

>> TCP OPEN или последовательность команд настройки соединения для перехода в активное или пассивное состояние OPEN **должна** быть дополнена возможностью настройки.

>> реализация TCP-AO **должна** разрешать изменение набора MKT для исходящих соединений TCP (не CLOSED).

MKT, связанные с соединением, должны быть доступные для подтверждения, включая возможность чтения этих MKT.

>> TCP STATUS **следует** дополнить для обеспечения возможности читать MKT текущего или ожидающего соединения (для подтверждения).

Отправителям может потребоваться возможность определять смену исходящего MKT (KeyID) или указание нового предпочтительного MKT (RnextKeyID). Эти изменения сразу влияют на все последующие исходящие сегменты.

>> TCP SEND или последовательность команд, приводящая к состоянию SEND, **должна** быть дополнена так, чтобы можно было указать предпочтительный MKT для исходящих (current_key) и/или входящих (rnext_key) сегментов.

Может оказаться полезной смена активного исходящего MKT (current_key) даже при отсутствии переданных данных, что можно выполнить путем отправки буфера нулевого размера или использования интерфейса без передачи (non-send, например, опции сокета в Unix) в зависимости от реализации.

Полезно также указывать полученные в недавнем сегменте значения KeyID и RnextKeyID. Хотя таких значений может быть множество, их быстрая смена не предполагается, поэтому любой свежей выборки должно быть достаточно.

>> TCP RECEIVE или последовательность команд, приводящая к RECEIVE, **должна** быть дополнена так, чтобы значения KeyID и RNextKeyID из недавно принятого сегмента были доступны пользователю по отдельному каналу (например, как дополнительный параметр RECEIVE или через вызов STATUS).

7.2. Состояния и переходы TCP

TCP включает состояния LISTEN, SYN-SENT, SYN-RECEIVED, ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT, CLOSED.

>> MKT **можно** связать с любым состоянием TCP.

7.3. Сегменты TCP

TCP включает сегменты управления (с флагом SYN, FIN, RST) и данных (без флагов SYN, FIN, RST), при этом сегменты управления могут включать данные (например, SYN).

>> Все сегменты TCP **должны** сравниваться с набором MKT на предмет соответствия идентификаторам соединений TCP.

>> Сегменты TCP, не прошедшие проверку TCP-AO, **должны** отбрасываться без уведомления отправителя.

>> Реализация TCP-AO **должна** разрешать настройку поведения сегментов с TCP-AO, не соответствующих MKT. Исходно по умолчанию такие соединения **следует** просто воспринимать. Если такое поведение нежелательно, можно включить MKT для сопоставления с такими соединениями или соединение может указывать обязательность TCP-AO. Кроме того, можно изменить конфигурацию для отбрасывания сегментов с опцией TCP-AO, не соответствующих MKT.

>> О событиях отбрасывания без уведомления отправителя **следует** сообщать пользователю в форме предупреждения, такую же индикацию **можно** принимать для восприятия без уведомления. Оба предупреждения (при их использовании) **должны** быть доступны через интерфейс STATUS. Любой из сигналов **может** быть асинхронным, но в этом случае сигналы **должны** ограничиваться по частоте. Любой из сигналов **может** записываться в системный журнал, но для частоты записи **следует** поддерживать возможность ограничения.

Обработка TCP-AO выполняется между интерфейсами TCP и IP. Для входящих сегментов это выполняется после проверки контрольной суммы TCP, для исходящих - перед расчетом контрольной суммы TCP.

Отметим, что использование TCP-AO на соединении не согласуется с TCP. Получатель отвечает за определение необходимости TCP-AO другими средствами (например, по отдельному каналу или с помощью протокола управления ключами) и исполнение этого требования.

7.4. Передача сегментов TCP

Приведенная ниже процедура описывает изменения TCP для поддержки вставки опции TCP-AO при отправке сегмента.

>> TCP-AO **должна** обрабатываться последней в заголовке TCP исходящего сегмента, поскольку в расчете MAC могут учитываться другие опции TCP.

1. Определение параметров соединения для сегмента.
   a. Если в сегменте имеется флаг SYN, этот сегмент является первым в соединении. Определяется ключ MKT для сегмента на основе пары сокетов этого сегмента.
      i. If Если нет соответствующего MKT, опция TCP-AO опускается и сегмент передается без нее.
      ii. При наличии соответствующего MKT устанавливаются нужные параметры на уровне соединения (раздел 4). Переход к п. 2.
   b. Если в сегменте нет флага SYN, проверяется использование TCP-AO для соединения и выбирается MKT в соответствии со значением current_key из параметров соединения (раздел 4). Переход к п. 2.
2. Использование заданных на уровне соединения параметров.

a. В заголовок TCP добавляется опция TCP-AO путем вставки полей Length и KeyID, соответствующих MKT, указанному current_key (используется current_key для MKT SendID как TCP-AO KeyID). Должным образом обновляется размер заголовка TCP.

b. Определяется SND.SNE в соответствии с параграфом 6.2.

c. Определяется подходящий ключ трафика (указанный current_key в соответствии с параграфом 6.1 и, возможно, кэшированный в TCB). Т. е. применяется send_SYN_traffic_key для сегментов SYN и send_other_traffic_key для других сегментов.

d. Определяется RNextKeyID в соответствии с указателем rnext_key и помещается в поле TCP-AO RNextKeyID (используется rnext_key для MKT RecvID как TCP-AO RNextKeyID[1]) в соответствии с параграфом 6.1.

e. Рассчитывается MAC с использованием MKT (и кэшированного ключа трафика) и данных из сегмента, как указано в параграфе 5.1.

f. Вставляется значение MAC в поле TCP-AO MAC.

g. Выполняется передача сегмента.

7.5. Прием сегментов TCP

Приведенная ниже процедура описывает изменения TCP для поддержки опции TCP-AO при получении сегмента.

>> Опция TCP-AO **должна** обрабатываться первой во входящем сегменте TCP, поскольку расчет MAC может включать другие опции TCP, которые могут измениться при обработке. Это также защищает другие опции TCP от влияния обманных сегментов или измененных данных заголовка.

>> Проверки TCP-AO **должны** выполняться для всех входящих сегментов SYN, чтобы избежать SYN без требуемой опции TCP-AO. Для других сегментов можно использовать кэшированные в TCB сведения о применении TCP-AO.

1. Отыскиваются параметры соединения для сегмента.

   a. Если в сегменте установлен флаг SYN, этот сегмент является первым в соединении. Отыскивается ключ MKT для этого сегмента по паре сокетов и TCP-AO KeyID, соответствующему идентификатору соединения TCP и RecvID для MKT.

      i. При отсутствии нужного MKT опция TCP-AO удаляется из сегмента и обработка продолжается. Это предполагает, что соединения, не соответствующие MKT воспринимаются без уведомления, как указано в параграфе 7.3.

      ii. При наличии соответствующего MKT устанавливаются нужные параметры соединения (раздел 4). Переход к п. 2.

2. Использование заданных на уровне соединения параметров.

   a. Проверка соответствия TCP-AO Length размеру, указанному MKT.

      i. Если размеры отличаются, сегмент отбрасывается без уведомления. Событие записывается в журнал и/или выдается предупреждение в соответствии с параграфом 7.3.

   b. Определяется RCV.SNE для сегмента в соответствии с параграфом 6.2.

   c. Определяется ключ трафика для сегмента из MKT в соответствии с параграфом 5.1 (вероятно кэшированный в TCB). Т. е. применяется receive_SYN_traffic_key для сегментов SYN и receive_other_traffic_key - для остальных сегментов.

   d. Рассчитывается MAC для сегмента с использованием MKT (и выведенного из него ключа трафика) и частей сегмента, как указано в параграфе 5.1.

      i. Если полученное значение отличается от поля TCP-AO MAC, сегмент отбрасывается без уведомления. Событие записывается в журнал и/или выдается предупреждение в соответствии с параграфом 7.3.

   e. Сравнивается полученное значение RNextKeyID с активным исходящим KeyID (current_key MKT SendID).

      i. При совпадении дополнительные действия не нужны.

      ii. Если значения различаются, проверяется готовность RNextKeyID MKT.

         1. Если MKT, соответствующий паре сокетов и RNextKeyID недоступен, дальнейших действий не требуется (RNextKeyID принимающей стороны должен соответствовать SendID в MKT).

         2. Если доступен MKT, соответствующий паре сегментов и RNextKeyID:

            a. Для current_key устанавливается RNextKeyID MKT.

   f. Выполняется обработка сегмента TCP.

Предполагается, что реализации TCP-AO проверяют поле Length в сегменте до расчета MAC для снижения издержек, которые могут быть вызваны поддельными сегментами с недействительными полями TCP-AO.

Дополнительное снижение издержек, связанных с проверкой MAC, может обеспечиваться алгоритмами MAC, например, за счет алгоритма расчета, добавляющего фиксированное значение в начало учитываемой части, и соответствующего алгоритма, проверяющего наличие этого значения в самом начале процесса. Такая оптимизация

[1] В исходном документе ошибочно указано KeyID. См. https://www.rfc-editor.org/errata/eid5961. *Прим. перев.*

будет включаться в спецификацию алгоритма MAC и ее не нужно радавать явно в TCP-AO. Отметим, что KeyID не может применяться для проверки соединения как такового, поскольку это значение не предполагается случайным.

7.6. Влияние на размер заголовка TCP

TCP-AO с изначально заданными 96-битовыми MAC добавляет в заголовок TCP 16 байтов [RFC5926], что на 2 байта меньше размера опции TCP MD5 (18 байтов).

Отметим, что пространство опций TCP наиболее критично для сегментов SYN, поскольку флаги в них могут увеличивать пространство опций в других сегментах. TCP игнорирует неизвестные сегменты, поэтому нет возможности расширить пространство опций SYN без возникновения несовместимости с прежними версиями.

4-битовое поле Data Offset в заголовке TCP требует, чтобы заголовок и опции занимали не более 60 байтов (15 слов по 32 бита) от начала заголовка, фиксированная часть которого составляет 20-байтов. Это оставляет для опций 40 байтов заголовка, из которых 15 предполагаются занятыми в текущих реализациях (см. ниже), и доступно лишь 25 байтов. TCP-AO занимает 16 байтов, оставляя 9 для дополнительных опций SYN (в от реализации до 2 байтов может тратиться на заполнение).

- Разрешение SACK (2 байта) [RFC2018][RFC3517].
- Временные метки (10 байтов) [RFC1323].
- Масштабирование окна (3 байта) [RFC1323].
- Максимальный размер сегмента (4 байта) [RFC793][1].

После SYN в текущих реализациях TCP предполагаются опции:

- SACK (10 байтов) [RFC2018][RFC3517] (18 байтов для D-SACK [RFC2883]);
- Временные метки (10 байтов) [RFC1323].

TCP-AO продолжает занимать 16 байтов в сегментах без SYN, оставляя 24 байта для других опций, из которых 10 байтов занимают временные метки. В результате остаются 14 байтов, из которых десять служат для одного блока SACK. При использовании 2 блоков SACK (например, для обработки D-SACK), требуется сокращать TCP-AO MAC для размещения дополнительного блока SACK (т. е. для 18 байтов опции D-SACK, являющейся вариантом SACK) [RFC2883].Отметим, что D-SACK не поддерживается с TCP MD5 при наличии временных меток, поскольку размер MAC в TCP MD5 фиксирован и не оставляет достаточно места.

Хотя пространство опций TCP ограничено, предполагается, что TCP-AO согласуется с желанием аутентифицировать TCP на уровне соединений с целями, для которых была предназначена опция TCP MD5.

7.7. Обработка сброса соединений

TCP-AO позволяет обмен сегментами сброса TCP (RST) при условии, что обе стороны находятся в допустимом (valid) состоянии соединения. После установки такого состояния в случае перезагрузки одной из сторон TCP-AO не позволяет механизму TCP RST сбросить (очистить) прежнее состояние на другой стороне. Это обусловлено тем, что перезагруженная сторона теряет состояние соединения и, как следствие, ключи трафика.

Важно, чтобы реализации были способны обнаруживать избыточные соединения TCP в такой конфигурации и устранять их при необходимости для экономии памяти [Ba10]. Для защиты от накопления таких состояний имеется ряд рекомендаций, приведенных ниже

>> В соединениях с TCP-AO **следует** применять также TCP keepalive [RFC1122].

Использование пакетов поддержки TCP (keepalive) гарантирует, что соединения с утраченными ключами будут прерываться за конечное время. Аналогичный эффект может быть обеспечен на уровне приложения, например, с помощью BGP keepalive [RFC4271]. Любой тип сообщений keepalive обеспечит очистку состояния TCP в таких случаях. Другой вариант, разрешающий принимать RST без аутентификации, имеет существенную уязвимость, которую можно предотвратить с помощью TCP-AO.

Сообщения keepalive обеспечивают сброс соединений при перезагрузке, но это может негативно влиять на некоторые протоколы. В частности, BGP плохо реагирует на такие разрывы соединений, даже при использовании BGP keepalive, поэтому была разработана специальная процедура «аккуратной перезагрузки» (graceful restart) [RFC4724], которая была затем расширена для BGP с MPLS [RFC4781]. В соответствии с этим:

>> соединениям BGP **следует** требовать поддержки graceful restart при использовании TCP-AO.

Понятно, что такая поддержка возможна (желателина) не во всех случаях, поэтому предлагается иной вариант.

>> При перезапуске BGP без поддержки graceful restart, но с использованием TCP-AO, обеим сторонам соединения **следует** сохранять ключи трафика и восстанавливать их после перезагрузки, а также **следует** ограничивать влияние такого сохранения и восстановления ключей на производительность.

7.8. Обработка ICMP

Протокол TCP можно атаковать по основному каналу с помощью сегментов TCP или по отдельному (out of band) с использованием ICMP. Пакеты ICMP невозможно защитить с помощью механизмов TCP-AO, однако прямую защиту сигнализации ICMP не может обеспечить как TCP-AO, так и IPsec. В TCP-AO даются конкретные рекомендации по обработке некоторых пакетов ICMP в дополнение к требуемым IPsec; это обусловлено работой TCP-AO в контексте соединения TCP.

IPsec включает рекомендации по отбрасыванию ICMP в определенном контексте и требования аутентификации конечных точек [RFC4301]. Имеются и другие механизмы, снижающие влияние атак ICMP на счет дополнительной проверки содержимого ICMP и смены влияния некоторых сообщений на основе состояния TCP, но они не

[1]Эта строка отсутствует в исходном документе. См. https://www.rfc-editor.org/errata/eid4365. *Прим. перев.*

обеспечивают уровень проверки подлинности ICMP, который TCP-AO предоставляет TCP [Go10]. В результате рекомендуется сдержанный подход к восприятию сообщений ICMP, как описано в [Go10].

>> Реализация TCP-AO по умолчанию **должна** игнорировать входящие сообщения ICMPv4 типа 3 (адресат недоступен) с кодами 2-4 (протокол недоступен, порт недоступен, требуется фрагментация - «жесткие ошибки»), и ICMPv6 типа 1 (адресат недоступен) с кодом 1 (административно запрещено) и 4 (порт недоступен), предназначенные для соединений в синхронизированных состояниях (ESTABLISHED, FIN-WAIT-1, FIN- WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT), которые соответствуют MKT.

>> Реализации TCP-AO **следует** разрешать настройку игнорирования таких ICMP на уровне соединения.

Реализации TCP-AO **следует** поддерживать меры защиты от сообщений ICMP о слишком больших пакетах (packet too big), некоторые примеры которых рассматриваются в [Go10].

>> Реализации **следует** разрешать запись событий игнорирования ICMP в системный журнал.

Это влияет лишь на сообщения ICMP, которые в настоящее время требуют жестких мер (hard error), связанных с разрывом соединения TCP [RFC1122]. Рекомендации аналогичны мерам обработки таких сообщений в IPsec, как указано в [RFC4301].

8. Отмена TCP MD5 и взаимодействие с устаревшими системами

TCP-AO отменяет опцию TCP MD5.

>> Реализации TCP, поддерживающие TCP MD5, **должны** поддерживать TCP-AO.

Системы, реализующие лишь TCP MD5, считаются устаревшими, и их следует обновлять по мере возможностей. Для поддержки взаимодействия с такими системами до их обновления следует выполнять приведенные ниже требования.

>> TCP MD5 **следует** поддерживать там, где нужно взаимодействие с устаревшими системами.

>> Система, поддерживающая TCP-AO и TCP MD5, **должна** применять TCP-AO для соединений, если это поддерживает партнер. В противном случае **можно** использовать TCP MD5.

>> Реализации TCP **недопустимо** применять TCP-AO и TCP MD5 на одном соединении TCP, но **можно** поддерживать одновременно TCP-AO и TCP MD5 для разных соединений (в частности, для работы с устаревшими системами TCP MD5).

Значение Kind явно указывает в сегментах использование TCP-AO или TCP MD5 на конкретном соединении TCP.

Можно расширить MKT для поддержки TCP MD5, но использование MKT не описано в RFC 2385.

Для соединения можно требовать TCP-AO или TCP MD5, но не то и другое сразу. Если конечная точка настроена требовать TCP MD5 на соединении, она должна включать эту опцию во все исходящие сегменты и проверять ее во входящих [RFC2385]. Требования TCP MD5 запрещают использование обеих опций на одном соединении, например, для передачи решения о выборе другой стороне.

9. Взаимодействие с промежуточными устройствами

TCP-AO может взаимодействовать с промежуточными устройствами в зависимости от их поведения [RFC3234]. Некоторые промежуточные устройства меняют опции TCP (такие как TCP-AO) напрямую или информацию, включенную в расчет TCP-AO MAC. Это может мешать работе TCP-AO, особенно при изменении данных, защищенных опцией.

9.1. Взаимодействие с устройствами без NAT/NAPT

TCP-AO поддерживает промежуточные устройства, не меняющие адреса IP или порты в сегментах. Такие устройства могут менять опции TCP и в этом случае нужно настраивать TCP-AO на игнорирование всех опций при расчете MAC для проходящих через такие устройства соединений.

Отметим, что игнорирование опций TCP может снижать уровень защиты, т. е. опции TCP можно будет незаметно изменить в пути, что открывает путь для атак. В зависимости от изменений (например, временных меток) может быть снижена эффективность TCP или нарушена корректность операций (например, при изменении размера окна). Эти уязвимости присущи лишь соединениям TCP, где для TCP-AO задано игнорирование опций TCP.

9.2. Взаимодействие с устройствами NAT/NAPT

TCP-AO не может естественным путем работать через устройства NAT/NAPT (Network Address Port Translation — трансляция адресов и портов), меняющие адреса IP и/или номера портов. Предполагается, что для прохождения через такие устройства могут потребоваться варианты имеющихся механизмов прохождения NAT/NAPT, например, инкапсуляция сегментов с защитой TCP-AO в другие транспортные сегменты (например, UDP), как принятов в IPsec [RFC2663][RFC3947]. Такие варианты могут быть приспособлены для работы с TCP-AO или можно использовать IPsec с прохождением через NAT вместо применения TCP-AO [RFC3947].

Другой вариант приспособления к NAT расширяет TCP-AO независимо от данной спецификации [To10].

10. Оценка выполнения требований

TCP-AO соответствует всем текущим требованиям к пересмотру TCP MD5, как показано ниже [Ed07].

1. Защищаемые элементы.

   Решению, заменяющему TCP MD5, следует защищать (проверять подлинность) перечисленные ниже элементы (выполнено, см. параграф 5.1).

   a. Псевдозаголовок IP, включая IPv4 и IPv6.

Отметим, что частичное покрытие не допускается, поскольку адреса IP определяют соединение. Если адреса можно согласовать через NAT/NAPT, отправитель может рассчитать MAC по полученным адресам, в остальных случаях требуется организация туннеля, как отмечено в параграфе 9.2.

b. Заголовок TCP.

Отметим, что частичное покрытие не допускается, поскольку порты определяют соединение. Если порты можно согласовать через NAT/NAPT, отправитель может рассчитать MAC по полученным номера, в остальных случаях требуется организация туннеля, как отмечено в параграфе 9.2.

c. Опции TCP.

TCP-AO разрешает исключать опции TCP из расчета MAC для работы через промежуточные устройства, меняющие опции (но не TCP-AO), как описано в раздел 9.

d. Данные TCP.

2. Требования к структуре опции.

Решению по замене TCP MD5 следует выполнять приведенные ниже требования к структуре опции (выполнено, см. параграф 5.1).

a. Конфиденциальность.

Опции следует не раскрывать без нужды информацию о механизме TCP-AO. Обеспечиваемая этим дополнительная защита может не иметь большого значения, зато помогает сократить размер опции. TCP-AO раскрывает лишь идентификаторы MKT, MAC и размер опции в линии. Отметим, что укороченные MAC можно скрыть, указав в размере опции полное значение и задав меньший размер MAC (это эквивалентно другому алгоритму MAC и указывается в MKT). См. параграф 2.2.

b. Разрешение на уровне соединения.

Опцию не следует требовать в каждом соединении, ее следует разрешать на уровне соединения.

Это поддерживается за счет возможности установки MKT, соответствующих лишь некоторым соединениям. Для соединений, не соответствующих ни одному MKT, применять TCP-AO не требуется. Кроме того, входящие сегменты с TCP-AO не отбрасываются исключительно по присутствию опции, не соответствующей ни одному MKT.

c. Требование использовать опцию.

Опции следует предоставлять возможность указания как требуемой для данного соединения. Это поддерживается за счет возможности установки набора MKT, который может соответствовать лишь некоторым соединениям. Опция TCP-AO требуется для соединений, соответствующих любому из MKT TCP-AO.

d. Стандартный анализ.

Опции следует быть легко анализируемой, т. е. не требовать дополнительного разбора и соответствовать стандартному формату опций RFC 793 (поддерживается, см. параграф 2.2).

e. Совместимость с Large Window и SACK.

Опции следует обеспечивать совместимость с опциями Large Window и SACK (поддерживается, см. параграф 7.6). Размер опции рассчитан на совместное использование с Large Window и SACK. См. также параграф 1.3, где указано, что TCP-AO на 2 байта короче опции TCP MD5 в принятом по умолчанию случае с 96-битовым MAC.

3. Криптографические требования.

Решению по замене TCP MD5 следует выполнять приведенные ниже криптографические требования.

a. Базовые значения по умолчанию.

Опции следует по умолчанию использовать значения, требуемые от всех реализаций. TCP-AO использует по умолчанию обязательный алгоритм, заданный в [RFC5926], как указано в параграфе 5.1.

b. Качественные алгоритмы.

Опции следует использовать алгоритмы, принятые сообществом безопасности, которые считаются достаточно защищенными. Следует избегать применения нестандартных или неопубликованных алгоритмов. TCP-AO использует MAC, указанные в [RFC5926], где заданы также функции KDF. Входные строки KDF соответствуют типовым [RFC5926].

c. Гибкость выбора алгоритмов.

Опции следует поддерживать отличающиеся от принятых по умолчанию алгоритмы для обеспечения гибкости с течением времени. TCP-AO позволяет применять любой желаемый алгоритм с учетом ограничений на размер опций TCP, как отмечено в параграфе 2.2. Использование набора MKT позволяет в разных соединениях выбирать свои алгоритмы как для MAC, так и для KDF.

d. Независимая от порядка доставки обработка.

Опции следует поддерживать обработку без зависимости от порядка доставки, т. е. следует разрешать обработку сегментов TCP, принятых с нарушением порядка, без переупорядочения. Это избавляет от необходимости восстанавливать порядок до обработки и влияния нарушения порядка доставки сегментов на опцию (поддерживается, см. параграфы 7.3 - 7.5). Отметим, что требуется предварительная обработка до TCP, поскольку сегмент TCP нельзя отбросить лишь на основании состояния соединения и проверки выхода за пределы окна. Многие из таких сегментов, хотя и отбрасываются, заставляют хост ответить

повтором последнего действительного ACK [RFC793]. В описании вывода номеров SNE на стороне получателя приведена демонстрация алгоритма, позволяющего избежать восстановления порядка (параграф 6.2).

e. Смена параметра защиты требует замены ключей.

Опции следует требовать смены MKT при изменении любого из параметров защиты. Это избавляет от необходимости координации состояние опции в процессе работы соединения, что типично для опций TCP. Это также позволяет разрешить реализации bump in the stack, не объединенные с реализацией TCP на конечной точке. Параметры меняются лишь при переходе на новый ключ MKT (раздел 3).

4. Требования к ключам.

Решению по замене TCP MD5 следует поддерживать установку ключей вручную, а также внешние системы автоматизированного управления ключами (например, протокол или иной механизм). Отметим, что TCP-AO не задает систему управления MKT.

a. Смена ключей в соединении.

Опции следует поддерживать смену ключей в соединении для устранения влияния на долгосрочные соединения. Это поддерживается за счет применения идентификаторов и множества MKT (раздел 3).

b. Эффективная смена ключей.

Опции следует поддерживать смену ключей в соединении без чрезмерного расхода вычислительных ресурсов. В частности, следует избегать необходимости проверки множества ключей для данного сегмента. Это поддерживается на основе применения KeyID (параграф 6.1).

c. Автоматизированная и ручная установка ключей.

Опции следует поддерживать ручную и автоматизированную установку ключей. Использование MKT позволяет задавать ключи вручную или автоматизированно (раздел 3). Это свойство усилено генерацией уникальных ключей для соединения, что позволяет применять заданные вручную MKT с автоматически генерируемыми ключами трафика (параграф 5.2).

d. Независимость от управления ключами.

Опции не следует предполагать или требовать наличия системы управления ключами. Это обеспечивается за счет поддержки множества MKT (раздел 3).

5. Ожидаемые ограничения.

Решению по замене TCP MD5 следует поддерживать типовые методы защиты.

a. Сокрытие отказов при проверке.

Получение сегментов, не прошедших проверку подлинности, не должно приводить к видимым извне действиям, недопустимо изменять в таких случаях состояние, а события следует записывать в системный журнал (поддерживается, см. параграфы 7.3 - 7.5).

b. Не более одной опции на сегмент.

В сегменте разрешена лишь 1 опция аутентификации (поддерживается в требованиях, см. параграф 2.2).

c. Проверять на выходе все или ничего.

Сегменты из соединения TCP должны проверяться все или не проверяться совсем (поддерживается, см. параграф 7.4).

d. На входе проверять все.

Входящие сегменты в соединении TCP всегда проверяются на предмет наличия и пригодности опции аутентификации (поддерживается, см. параграф 7.5).

e. Отсутствие взаимодействия с TCP MD5.

Использование этой опции для данного соединения не должно препятствовать применению TCP MD5 на других соединениях, например, при работе с устаревшими системами (поддерживается, см. раздел 8).

f. «Жесткое» отбрасывание ICMP.

Опции следует разрешать отбрасывать некоторые сообщения ICMP, особенно типа 3 (destination unreachable) с кодами 2-4 (transport protocol unreachable, port unreachable, fragmentation needed and IP DF field set), говорящие об отказе конечной точки от взаимодействия (поддерживается, см. параграф 7.8).

g. Поддержка семантики соединений TCP, где лишь пара сокетов связывает ассоциацию TCP и все параметры ее защиты (поддерживается, см. разделы 3 и 9).

11. Вопросы безопасности

Применение TCP-AO будет влиять на производительность, подобно TCP MD5 или IPsec. Соединения, для которых известно применение TCP-AO, могут быть атакованы сегментами с недействительными MAC. Атакующему нужно знать лишь идентификатор соединения ID и значение TCP-AO Length для воздействия на возможности хоста обрабатывать трафик. Это похоже на уязвимость IPsec к атакам в пути, где адреса IP и SPI будут видны. В IPsec доступно все пространство SPI (32 бита), тогда как протоколы маршрутизации обычно могут произвольно менять лишь порт-источник (16 битов, но на деле имеется меньшее число случайных битов [La10]). В результате атакующему, не находящемуся на пути доставки проще передавать фиктивные сегменты TCP-AO, которые могут потребовать усилий по обработке на приемной стороне. Однако следует отметить, что между маршрутизаторами Internet произвольными могут быть

значения обоих портов (задаются по отдельному каналу), что будет представлять примерно такой же уровень защиты от подставных пакетов, как произвольные SPI.

TCP-AO, подобно TCP MD5, может запрещать сброс без организации соединения. Такой сброс обычно связан с отказом одного из партнеров, ответом на попытку соединения или передачей данных в устаревшее (stale) соединение. В любом случае восстанавливающая конечная точку может не иметь нужных ключей соединения (например, потеряны при отказе). Это может приводить к тайм-аутам, а не к ускоренному восстановлению после отказа. Рекомендации по снижению таких воздействий приведены в параграфе 7.7.

В TCP-AO нет возможности быстрого отклонения, например, при получении SYN-ACK без ожидаемого TCP-AO с быстым сбросом или разрывом соединения. Обычной реакцией TCP будет повтор или тайм-аут, которого следует ожидать, когда предусмотренный получатель не способен поддержать требуемый вариант TCP. Отказ от этого не предусмотрен, поскольку он открыл бы атакующим путь к прерыванию попыток соединения путем отправки фиктивных SYN-ACK без TCP-AO.

Опция TCP-AO нацелена на обеспечение защиты, подобной IPsec, но не для замены IPsec или IKE на более отказоустойчивую защиту или изощренное управления защитой. Опция TCP-AO предназначена для защиты от атак самого протокола TCP, которую не могут предоставить такие протоколы как TLS, sBGP/soBGP и др. Подобно IPsec, TCP-AO не решает до конца вопросы защиты от ICMP-атак на TCP, но ограничивает влияние ICMP, как отмечено в параграфе 7.8.

TCP-AO включает идентификаторы соединений TCP (пара сокетов) в расчет MAC. Это предохраняет обновременные соединения, использующие один MKT (по любой причине) от атак с перекрестным трафиком, когда сегменты с одной парой сокетов перенаправляются на другую пару. При использовании несколькими соединениями одного MKT, полезно знать, что сегменты для одной пары не могут (умышленно или ненароком) изменены при передаче и аутентифицированы для другой пары. Это требование задает дополнительную нагрузку, связанную с уникальностью MKT в конечной системе и, возможно, среди разных конечных систем. Хотя вероятность атак невелика, защита, обеспечиваемая включением принятого идентификатора гарантирует его учет в MAC и не усложняет чрезмерно расчет MAC и управление MKT.

Использование любого алгоритма защиты может сделать возможными DoS-атаки[1] на CPU, когда злоумышленник передает случайные фиктивные сегменты, на обработку которых получатель затрачивает много ресурсов CPU. В IPsec ослабление таких атак обеспечивается использованием больших полей SPI (Security Parameter Index -индекс параметров защиты) и порядковых номеров для частичной проверки сегментов до того, как будут затрачены ресурсы CPU на проверку ICV (Integrity Check Value - код контроля целостности). В TCP-AO пара сокетов выполняет большинство функций SPI, а порядковый номер IPsec, служащий для предотвращения replay-атак, не требуется по причине наличия порядкового номера TCP, служащего для восстановления порядка принятых сегментов (при условии, что порядковый номер не зациклен, что обеспечивается добавлением в TCP-AO номеров SNE, описанный в параграфе 6.2). TCP уже защищает себя от повторов аутентичных сегментов данных и управления (например, биты SYN, FIN, ACK), но даже аутентичные повторные пакеты могут влиять на контроль перегрузок в TCP [Sa99]. TCP-AO не защищает контроль перегрузок TCP от последней формы атак из-за громоздкости применения порядковых номеров окна защиты в дополнение к обычным номерам TCP. При желательности такой защиты следует применять IPsec.

Кроме того, бесполезно проверять порядковый номер TCP до выполнения расчетов TCP-AO, поскольку сегменты за пределами окна все равно вызывают корректные действия протокола TCP (например, повтор ACK) [RFC793]. Также бесполезно добавлять отдельное поле порядкового номера в TCP-AO, поскольку это может привести к изменению поведения TCP даже для корректных сегментов.

12. Взаимодействие с IANA

Опции аутентификации TCP (TCP-AO) в IANA был назначен номер 29.

Этот документ не задает новых пространств имен.

Задание алгоритмов MAC и KDF для TCP-AO вынесено в отдельный документ [RFC5926].

13. Литература

13.1. Нормативные документы

[RFC793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.

[RFC2018] Mathis, M., Mahdavi, J., Floyd, S., and A. Romanow, "TCP Selective Acknowledgment Options", RFC 2018, October 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2385] Heffernan, A., "Protection of BGP Sessions via the TCP MD5 Signature Option", RFC 2385, August 1998.

[RFC2403] Madson, C. and R. Glenn, "The Use of HMAC-MD5-96 within ESP and AH", RFC 2403, November 1998.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460[2], December 1998.

[RFC2883] Floyd, S., Mahdavi, J., Mathis, M., and M. Podolsky, "An Extension to the Selective Acknowledgement (SACK) Option for TCP", RFC 2883, July 2000.

[RFC3517] Blanton, E., Allman, M., Fall, K., and L. Wang, "A Conservative Selective Acknowledgment (SACK)-based Loss Recovery Algorithm for TCP", RFC 3517[3], April 2003.

[1] Denial-of-Service - отказ в обслуживании.
[2] Современная спецификация IPv6 задана в RFC 8200. *Прим. перев.*
[3] Заменен RFC 6675. *Прим. перев.*

[RFC4306] Kaufman, C., Ed., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005.

[RFC4724] Sangli, S., Chen, E., Fernando, R., Scudder, J., and Y. Rekhter, "Graceful Restart Mechanism for BGP", RFC 4724, January 2007.

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.

[RFC4781] Rekhter, Y. and R. Aggarwal, "Graceful Restart Mechanism for BGP with MPLS", RFC 4781, January 2007.

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

13.2. Дополнительная литература

[Ba10] Bashyam, M., Jethanandani, M., and A. Ramaiah "Clarification of sender behaviour in persist condition", Work in Progress[1], January 2010.

[Bo07] Bonica, R., Weis, B., Viswanathan, S., Lange, A., and O. Wheeler, "Authentication for TCP-based Routing and Management Protocols", Work in Progress, February 2007.

[Bo09] Borman, D., "TCP Options and MSS", Work in Progress, July 2009[2].

[Ed07] Eddy, W., Ed., Bellovin, S., Touch, J., and R. Bonica, "Problem Statement and Requirements for a TCP Authentication Option", Work in Progress, July 2007.

[Go10] Gont, F., "ICMP Attacks against TCP", Work in Progress[3], March 2010.

[La10] Larsen, M. and F. Gont, "Transport Protocol Port Randomization Recommendations", Work in Progress[4], April 2010.

[Le09] Lepinski, M. and S. Kent, "An Infrastructure to Support Secure Internet Routing", Work in Progress[5], October 2009.

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

[RFC1323] Jacobson, V., Braden, R., and D. Borman, "TCP Extensions for High Performance", RFC 1323, May 1992.

[RFC1948] Bellovin, S., "Defending Against Sequence Number Attacks", RFC 1948, May 1996.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[RFC2663] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, August 1999.

[RFC3234] Carpenter, B. and S. Brim, "Middleboxes: Taxonomy and Issues", RFC 3234, February 2002.

[RFC3562] Leech, M., "Key Management Considerations for the TCP MD5 Signature Option", RFC 3562, July 2003.

[RFC3947] Kivinen, T., Swander, B., Huttunen, A., and V. Volpe, "Negotiation of NAT-Traversal in the IKE", RFC 3947, January 2005.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC4808] Bellovin, S., "Key Change Strategies for TCP-MD5", RFC 4808, March 2007.

[RFC4953] Touch, J., "Defending TCP Against Spoofing Attacks", RFC 4953, July 2007.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246[6], August 2008.

[Sa99] Savage, S., N. Cardwell, D. Wetherall, T. Anderson, "TCP Congestion Control with a Misbehaving Receiver", ACM Computer Communications Review, V29, N5, pp71-78, October 1999.

[SDNS88] Secure Data Network Systems, "Security Protocol 4 (SP4)", Specification SDN.401, Revision 1.2, July 12, 1988.

[To07] Touch, J. and A. Mankin, "The TCP Simple Authentication Option", Work in Progress, July 2007.

[To10] Touch, J., "A TCP Authentication Option NAT Extension", Work in Progress[7], January 2010.

[Wa05] Wang, X., H. Yu, "How to break MD5 and other hash functions", Proc. IACR Eurocrypt 2005, Denmark, pp.19-35.

[We05] Weis, B., Appanna, C., McGrew, D., and A. Ramaiah, "TCP Message Authentication Code Option", Work in Progress, December 2005.

14. Благодарности

Этот документ был совместно разработан командой TCP Authentication Design (tcp-auth-dt), членами которой были (по алфавиту): Mark Allman, Steve Bellovin, Ron Bonica, Wes Eddy, Lars Eggert, Charlie Kaufman, Andrew Lange, Allison Mankin, Sandy Murphy, Joe Touch, Sriram Viswanathan, Brian Weis, Magnus Westerlund. Текст документа взят из предложения Joe Touch и Allison Mankin [To07] (изначально июнь 2006 г.), которое было задумано как контрпредложение к пересмотру TCP MD5, предложенному в документе Ron Bonica, Brian Weis, Sriran Viswanathan, Andrew Lange, and Owen Wheeler [Bo07] (исходно сентябрь 2005 г.) и в документе Brian Weis [We05].

[1]Опубликована в RFC 6429. *Прим. перев.*
[2]Опубликована в RFC 6691. *Прим. перев.*
[3]Опубликована в RFC 5927. *Прим. перев.*
[4]Опубликована в RFC 6056. *Прим. перев.*
[5]Опубликована в RFC 6480. *Прим. перев.*
[6]В RFC 8446 представлена версия протокола 1.3. *Прим. перев.*
[7]Опубликована в RFC 6978. *Прим. перев.*

Russ Housley предложив управление кортежами первичных ключей на уровне L4 и прикладном. Steve Bellovin был за поле KeyID. Eric Rescorla предложил использовать начальные порядковые номера TCP ISN при расчете ключей трафика и SNE для предотвращения replay-атак, а Brian Weis расширил расчет путем включения идентификатора соединения полностью и предоставил детали расчета ключей трафика. Mark Allman, Wes Eddy, Lars Eggert, Ted Faber, Russ Housley, Gregory Lebovitz, Tim Polk, Eric Rescorla, Joe Touch и Brian Weis разработали механизм координации первичных ключей.

Alfred Hoenes, Charlie Kaufman, Adam Langley и многие другие члены рабочей группы TCPM WG предоставили существенные отклики для этого документа.

Документ изначально был подготовлен с использованием шаблона 2-Word-v2.0.template.dot.

Адреса авторов

Joe Touch

USC/ISI

4676 Admiralty Way

Marina del Rey, CA 90292-6695

U.S.A.

Phone: +1 (310) 448-9151

EMail: touch@isi.edu

URL: http://www.isi.edu/touch

Allison Mankin

Johns Hopkins Univ.

Baltimore, MD

U.S.A.

Phone: 1 301 728 7199

EMail: mankin@psg.com

URL: http://www.psg.com/~mankin/

Ronald P. Bonica

Juniper Networks

2251 Corporate Park Drive

Herndon, VA 20171

U.S.A.

EMail: rbonica@juniper.net

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru