Internet Engineering Task Force (IETF)
Request for Comments: 7413
Category: Experimental
ISSN: 2070-1721

Y. Cheng
J. Chu
S. Radhakrishnan
A. Jain
Google

December 2014

TCP Fast Open

Механизм TFO

Аннотация

Этот документ описывает экспериментальный механизм TCP, названный TCP Fast Open (TFO). TFO позволяет передавать данные в пакетах SYN и SYN-ACK и воспринимать их на приемной стороне в процессе начального согласования соединения. Это позволяет сэкономить один интервал кругового обхода (round-trip time или RTT) по сравнению со стандартным TCP, где применяется трехэтапное согласование (three-way handshake или 3WHS) до того, как может начаться обмен данными. Однако TFO отступает от стандартной семантики TCP, поскольку данные в SYN при некоторых обстоятельствах допускают повторное использование (replay). Приложениям не следует использовать TFO, если они не готовы к таким ситуациям. Этот вопрос рассмотрен в разделе 6. Применимость TFO.

Статус документа

Документ не является спецификацией стандарта (Internet Standards Track) и публикуется для проверки, экспериментальной реализации и оценки.

Документ определяет экспериментальный протокол (Experimental Protocol) для сообщества Internet. Документ является результатом работы IETF[1] и представляет согласованный взгляд сообщества IETF. Документ прошёл открытое обсуждение и был одобрен для публикации IESG[2]. Не все одобренные IESG документы претендуют на статус того или иного стандарта Internet (см. раздел 2 в RFC 5741).

Информация о текущем статусе документа, найденных ошибках и способах обратной связи доступна по ссылке http://www.rfc-editor.org/info/rfc7413.

Оглавление

[1]Internet Engineering Task Force - комиссия по решению инженерных задач Internet.
[2]Internet Engineering Steering Group - комиссия по инженерным разработкам Internet.

1. Введение

TFO является экспериментальным обновлением TCP, которое позволяет обмен данными на этапе согласования соединения TCP. Этот документ описывает решение, которое позволяет приложениям сэкономить один интервал кругового обхода без серьезного влияния на безопасность. Основой TFO является значение security cookie, используемое серверной стороной для проверки подлинности (аутентификации) клиента, инициирующего соединение TFO. Документ описывает детали обмена данными в процессе начального согласования TCP, протокол для TFO cookie, возможные новые проблемы безопасности и их ослабление, а также новых интерфейс API для сокетов.

Мотивом разработки TFO послужили потребности современных Web-приложений. Протокол TCP разрешает обмен данными лишь после завершения трехэтапного согласования (3WHS) [RFC793], что добавляет задержку в один интервал RTT. Для коротких обменов Web этот дополнительный интервал RTT составляет существенную часть общей задержки в сети даже при широком использовании постоянных соединений HTTP. Например, браузер Chrome [Chrome] сохраняет бездействующие соединения TCP до 5 минут, но 35% запросов HTTP выполняются в новых соединениях TCP [RCCJR11]. Для таких приложений Web (и подобных им) включение данных в пакет SYN может значительно снизить задержку. Далее описано решение проблем, возникающих в результате такого обмена данными.

1.1. Терминология

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].

TFO является сокращением для TCP Fast Open, клиентом называется активная (открывающая) сторона соединения TCP, сервером - пассивная.

2. Данные в SYN

Стандартный протокол TCP позволяет передавать данные в пакетах SYN ([RFC793], параграф 3.4), но запрещает доставлять их приложению до завершения 3WHS. Это связано с тем, что начальное согласование TCP служит для захвата старых и продублированных пакетов SYN.

Чтобы разрешить приложениям обмен данными во время согласования TCP это ограничение снято в TFO и данные из пакетов SYN разрешается доставлять приложению. Это изменение семантики TCP вызывает две проблемы (рассматриваются ниже), которые не позволяют применять TFO для некоторых приложений.

Поэтому реализациям TCP **недопустимо** использовать TFO по умолчанию и можно применять TFO лишь при явном запросе приложения или настройке на уровне сервисного порта. Приложениям нужно перед использованием проверить применимость TFO, как описано в разделе 6.

2.1. Смягчение семантики TCP в части дубликатов SYN

TFO разрешает доставлять данные приложению до завершения 3WHS, открывая себя в плане целостности данных в перечисленных ниже ситуациях:

a) принимающий хост получает данные в дубликатах SYN после того, как он забыл об их получении в исходном SYN (например, в результате перезагрузки);

b) дубликат получен после того, как организованное исходным пакетом SYN соединение было закрыто по инициативе отправителя (получатель не защищен состоянием TIME-WAIT 2 MSL).

В отмененном ныне T/TCP [RFC1644] (отменен [RFC6247]) предпринималась попытка решить эти проблемы. Попытка оказалась неудачной и решение не было развернуто по причине уязвимостей, описанных в разделе 8. Смежные работы. Вместо попытки перехвата всех сомнительных пакетов SYN, чтобы сделать TFO полностью совместимым с семантикой TCP, принято решение воспринимать старые пакеты SYN с данными, т. е. ограничить применение TFO классом приложений, устойчивых к дубликатам пакетов SYN с данными (6. Применимость TFO). Предполагается, что это обеспечит верный компромисс между сложностью и применимостью.

2.2. Пакеты SYN с подставными адресами IP

Стандартный протокол TCP подвержен атакам SYN flood [RFC4987], поскольку пакеты SYN с подставными IP-адресами отправителей могут легко переполнить небольшую очередь прослушивающего приложения, что ведет к полной блокировке порта службы.

TFO делает еще один шаг вперед, позволяя серверной стороне TCP передавать данные приложению до завершения 3WHS. Это открывает серьезную уязвимость. Приложение, обслуживающее порт, на котором включен TFO, может потреблять значительные ресурсы CPU и памяти на обработку запросов и создание откликов. Если отклик существенно больше запроса, атакующий будет способен организовать атаку с усиленным отражением, направленную на жертву за пределами сервера TFO.

Имеется много методов ослабления атак SYN flood, описанных в [RFC4987], но, к сожалению, ни один из них не подходит для TFO. Здесь предлагается использовать предоставляемые сервером cookie для смягчения новых уязвимостей (раздел 3) и представлена оценка эффективности такой защиты (раздел 7).

3. Обзор протокола

Важной частью TFO является Fast Open Cookie (cookie) - код аутентификации сообщения (message authentication code или MAC), создаваемый сервером. Клиент запрашивает cookie в одном обычном соединении TCP и затем использует код для будущих соединений TCP при обмене данными в процессе 3WHS.

Запрос Fast Open Cookie.

1. Клиент передает пакет SYN с опцией Fast Open и пустым полем cookie для запроса кода.
2. Сервер генерирует код cookie и возвращает его в опции Fast Open пакета SYN-ACK.
3. Клиент кэширует cookie для будущих соединений TCP Fast Open (см. ниже).

Выполнение процедуры TCP Fast Open.

1. Клиент передает пакет SYN с данными и cookie в опции Fast Open.
2. Сервер проверяет пригодность cookie.
   a. Если значение cookie действительно, сервер передает подтверждение SYN-ACK для SYN и данных. Затем сервер доставляет полученные данные приложению.
   b. В противном случае сервер отбрасывает данные и передает подтверждение SYN-ACK лишь для порядкового номера SYN.
3. Если сервер воспринимает данные из пакета SYN, он может передать данные отклика до завершения согласования. Максимальный объем передаваемых данных определяется контролем перегрузок TCP [RFC5681].
4. Клиент передает ACK с подтверждением SYN и данных сервера. Если данные клиента не были подтверждены, он повторяет их отправку в пакете ACK.
5. Остальная часть соединения выполняется как в обычном TCP. Клиент может повторять операции Fast Open после получения cookie (до истечения срока действия, заданного сервером). Таким образом, механизм TFO полезен для приложений, имеющих временную привязку соединений между клиентом и сервером.

Запрос Fast Open Cookie в соединении 1.

```
TCP A (клиент)                                        TCP B (сервер)
_____________                                         _____________
CLOSED                                                       LISTEN

#1 SYN-SENT       ----- <SYN,CookieOpt=NIL>  ---------->  SYN-RCVD
#2 ESTABLISHED    <---- <SYN,ACK,CookieOpt=C> ----------  SYN-RCVD
(кэширование cookie C)
```

Процедура TCP Fast Open в соединении 2.

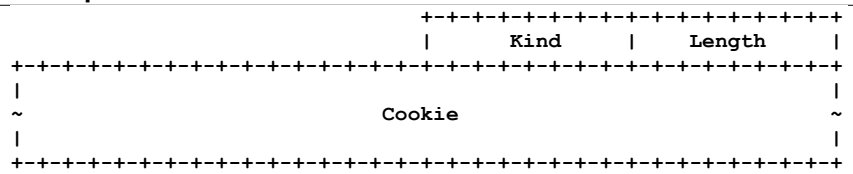
```
TCP A (клиент)                                        TCP B (сервер)
_____________                                         _____________
CLOSED                                                       LISTEN

#1 SYN-SENT       ----- <SYN=x,CookieOpt=C,DATA_A> ---->  SYN-RCVD
#2 ESTABLISHED    <---- <SYN=y,ACK=x+len(DATA_A)+1> ----  SYN-RCVD
#3 ESTABLISHED    <---- <ACK=x+len(DATA_A)+1,DATA_B>----  SYN-RCVD
#4 ESTABLISHED    ----- <ACK=y+1>--------------------> ESTABLISHED
#5 ESTABLISHED    --- <ACK=y+len(DATA_B)+1>----------> ESTABLISHED
```

4. Детали протокола

4.1. Fast Open Cookie

Fast Open Cookie служит для смягчения новых уязвимостей при разрешении обмена данных в процессе согласования соединений. Значением cookie является код MAC, генерируемый сервером. Клиент не анализирует это значение, а просто кэширует cookie и передает его серверу в последующих пакетах SYN при создании новых соединений. Сервер может прекратить действие cookie в любой момент с целью защиты.

4.1.1. Опция Fast Open



Kind
1 байт со значение 34.
Length
1 байт, задающий размер (от 6 до 18 байтов). Размер ограничен свободным пространством поля опций. Значение поля **должно** быть четным.
Cookie
0 или от 4 до 16 байтов (Length - 2).
Опция Fast Open служит для запроса или передачи Fast Open Cookie. Когде поле cookie отсутствует или пусто, опция используется клиентом для запроса cookie у сервера. При наличии cookie опция служит для передачи кода от сервера к клиенту или от клиента к серверу (процедура Fast Open).

Минимальный размер cookie составляет 4 байта. Хотя на рисунке показано выравнивание cookie по 32-битовым границам, такое выравнивание не требуется. Опции с некорректным значением Length или без флага SYN **должны** игнорироваться.

4.1.2. Обработка Cookie на сервере
Сервер отвечает за генерацию и проверку подлинности cookie. В качестве cookie **следует** использовать код MAC с приведенными ниже свойствами. Требование **следует** указано потому, что в некоторых случаях может применяться тривиальная генерация cookie (параграф 7.3).

1. Значение cookie аутентифицирует IP-адрес клиента (отправителя), который может быть IPv4 или IPv6.
2. Значения cookie могут создаваться только сервером, но не другими сторонами, включая клиентов.
3. Генерация и проверка выполняются достаточно быстро по сравнению с другой обработкой SYN и SYN-ACK.
4. Сервер может закодировать в cookie другую информацию и воспринимать от клиента в любой момент несколько значений cookie. Это зависит от реализации сервера и невидимо для клиента.
5. Для cookie устанавливается срок действия, причины этого рассмотрены в разделе 5. Вопросы безопасности. Это может быть периодическая смена ключа, используемого сервером для создания cookie, или включение временной метки при создании cookie.

   Для постепенного аннулирования cookie с течением времени сервер может реализовать ротацию ключей, используемых для создания и проверки cookie. Такой подход полезен для больших серверов, сохраняющих обновления ключей Fast Open. Конкретный механизм не задается, поскольку он зависит от реализации сервера.

Сервер поддерживает операции создания и проверки cookie:

- GetCookie(IP_Address) возвращает (новое) значение cookie;
- IsCookieValid(IP_Address, Cookie) проверяет пригодность cookie, т. е. соответствие IP-адресу клиента и срок действия.

Примером может реализации служить использование AES_128 для шифрования адресов IPv4 (с дополнением) или IPv6 и отсечки до 64 битов. Сервер может периодически обновлять ключ для завершения срока действия cookie. Ширование AES на современных процессорах выполняется всего на несколько сотен наносекунд [RCCJR11].

Если разрешается лишь одно значение cookie для адреса IP и сервер может регенерировать cookie независимо, лучшим вариантом проверки будет просто регенерация действительного cookie и сравнение с принятым значением. В этом случае при отрицательном результате проверки принятого cookie уже будет готово значение cookie для отправки клиенту.

4.1.3. Обработка Cookie клиентом
Клиент **должен** кэшировать cookie от сервера для последующих соединений Fast Open. Для многодомных клиентов cookie зависят от IP-адресов клиента и сервера, поэтому клиенту следует кэшировать не более одного (последнего) cookie для каждой пары IP-адресов сервера и клиента.

При кэшировании cookie рекомендуется кэшировать также анонсированный сервером максимальный размер сегмента (MSS). Клиент может кэшировать MSS от сервера для определения максимального объема данных, которые можно поместить в пакет SYN при следующих соединениях TFO. Такое кэширование полезно, поскольку при использовании Fast Open клиент передает данные в пакете SYN до того, как сервер анонсирует MSS в своем пакете SYN-ACK. Если клиент поместит в пакет SYN больше данных, чем сервер может воспринять, это вероятно потребует от клиента повторной передачи всех или части этих данных. Поэтому кэширование MSS может повысить производительность.

Без кэширования MSS объем данных в пакетах SYN ограничивается принятым по умолчанию значением MSS в 536 байта для IPv4 [RFC1122] и 1220 байтов для IPv6 [RFC2460]. Даже при выполнении клиентом этого требования при отправке SYN ясно, что получатель IPv4, анонсирующий MSS < 536 байтов, может получить сегмент большего размера.

Если кэшированное значение MSS больше типового размера (1460 байтов для IPv4 или 1440 для IPv6), избыточные данные в пакете SYN могут вызвать проблемы, которые сведут на нет преимущества Fast Open. Например, слишком большой пакет SYN может вызвать фрагментацию IP и ввести в заблуждение межсетевые экраны и промежуточные

устройства, что вызовет повторную передачу SYN и другие побочные эффекты. Поэтому клиент **может** ограничить кешируемое значение MSS до 1460 байтов для IPv4 или 1440 для IPv6.

4.1.3.1. Кэширование клиентом негативных откликов

Клиент **должен** кэшировать негативные отклики от сервера для предотвращения возможных отказов при соединениях. Негативные отклики включают отсутствие подтверждения от сервера данных в пакете SYN, сообщения ICMP об ошибках и (наиболее важно) полное отсутствие отклика SYN-ACK от сервера, т. е. тайм-аут в соединении. Последний случай вероятно обусловлен несовместимостью с промежуточными устройствами или полной блокировкой соединения межсетевым экраном после обработки пакета SYN с данными. Если клиент не реагирует на эти негативные отклики и продолжает попытки Fast Open, соединение с сервером может оказаться совсем невозможным.

Для любых негативных откликов клиенту **следует** отключить Fast Open на конкретном пути (адреса и порты отправителя и получателя) по меньшей мере на время. Поскольку TFO включается на уровне порта, а cookie не зависят от порта, клиенту следует включать в кэш и номера портов.

4.2. Протокол Fast Open

Одним из преобладающих требований к TFO является полная совместимость с имеющимися реализациями TCP на стороне клиентов и серверов.

Сервер поддерживает две переменных на каждый прослушиваемый сокет (IP-адрес и порт).

FastOpenEnabled
: По умолчанию режим отключен (off) и **должен** включаться приложением явно. При выключенном режиме сервер не выполняет связанных с TFO операций и **должен** игнорировать опции cookie.

PendingFastOpenRequests
: Задает отслеживание соединений TFO в состоянии SYN-RCVD. Если эта переменная превышает ранее установленный системный предел, сервер **должен** отключить TFO для всех новых запросов на соединение, пока PendingFastOpenRequests не станет меньше установленного в системе предела. Переменная служит для защиты от некоторых уязвимостей, рассматриваемых в разделе 5. Вопросы безопасности.

Сервер сохраняет флаг FastOpened на уровне соединения, чтобы пометить успешное использование TFO.

4.2.1. Запрос Fast Open Cookie

При любой попытке использовать TFO клиент **должен** сначала запросить у сервера cookie, как показано ниже.

1. Клиент передает пакет SYN с опцией Fast Open, где Length = 0 (пустое поле cookie).
2. Сервер отвечает пакетом SYN-ACK в соответствии с параграфом 4.1.2. Обработка Cookie на сервере. SYN-ACK может включать опцию Fast Open если сервер в данный момент поддерживает TFO на этом порту.
3. Если SYN-ACK включает опцию Fast Open с полем cookie, клиент меняет cookie и другую информацию в соответствии с параграфом 4.1.3. Обработка Cookie клиентом . В ином случае, если SYN-ACK обнаружен впервые и не является (ложным) повтором, клиент **может** удалить данные сервера из кэша cookie. Если SYN-ACK является ложным повтором, клиент ничего не делает с кэшем, по приведенным ниже причинам.

Сеть или серверы могут отбрасывать пакеты SYN или SYN-ACK с опцией cookie, что вызывает тайм-аут SYN или SYN-ACK. Клиентам и серверам **рекомендуется** в таком случае повторить передачу пакетов SYN и SYN-ACK без опции cookie. Это гарантирует прохождение соединению с запросом cookie и снизит задержку отброшенных пакетов SYN/SYN-ACK. Очевидным недостатком максимальной совместимости является то, что любое обычное отбрасывание SYN приведет к отказу cookie (хотя можно утверждать, что задержка передачи данных до завершения 3WHS обоснована, если пакет SYN отброшен в результате перегрузки). В следующем параграфе описана эвристика обнаружения таких отбрасываний, когда клиент получает пакет SYN-ACK.

Клиентам также **рекомендуется** записывать набор серверов, которые не отвечают на запросы cookie, и пытаться снова запрашивать cookie у них лишь по истечении некоторого времени.

4.2.2. TCP Fast Open

После получения клиентом cookie от нужного сервера он может организовывать последующие соединения TFO, пока не завершится срок действия cookie, заданный сервером.

Клиент передает SYN.

Для организации соединения TFO клиент **должен** иметь cookie от сервера.

1. Передается пакет SYN.
   a. Если в пакете SYN нет места для опции Fast Open, соединение TFO прерывается и используется обычное согласование 3WHS.
   b. В ином случае в заголовок включается опция Fast Open с полученным от сервера значением cookie. Включаются также данные размером не более кэшированного значения MSS от сервера или принятых по умолчанию 536 байтов.
2. Переход в состояние SYN-SENT и обновление SND.NXT для включения данных.

При работе с сетью и серверами, отбрасывающими пакеты SYN с данными или неизвестными опциями, когда срабатывает таймер SYN, клиенту **следует** повторить передачу пакета SYN без опции Fast Open.

Сервер получает SYN и отвечает пакетом SYN-ACK.

При получении пакета SYN с опцией Fast Open сервер выполняет указанные ниже действия.

1. Инициализируется и сбрасывается локальный флаг FastOpened. Если FastOpenEnabled = false, переход к п. 5.

2. Если PendingFastOpenRequests превышает системный предел, переход к п. 5.
3. Если IsCookieValid() (параграф 4.1.2) возвращает false, переход к п. 5.
4. Данные буферизуются и уведомляется приложение. Устанавливается флаг FastOpened и инкрементируется PendingFastOpenRequests.
5. Передается пакет SYN-ACK, который **может** включать опцию Fast Open. При установленном флаге FastOpened пакет подтверждает SYN и полученные данные, а противном случае - только SYN. Сервер **может** включить данные отклика в пакет SYN-ACK, если они уже готовы. Некоторые приложения предпочитают задерживать SYN-ACK, позволяя приложению обработать запрос для создания отклика, но это остается за реализацией.
6. Переход в состояние SYN-RCVD. Если установлен флаг FastOpened, сервер **должен** следовать [RFC5681] (основан на [RFC3390]) для установки начального окна насыщения, чтобы передать дополнительные пакеты.

При срабатывании таймера SYN-ACK серверу **следует** повторно передать сегмент SYN-ACK без данных и опций Fast Open по причинам совместимости.

Особым случаем является одновременная организация соединения, когда получателем SYN является клиент в состоянии SYN-SENT. Протокол не меняется, поскольку [RFC793] уже поддерживает данные в SYN и одновременную организацию соединения. Но сокет клиента может иметь данные, доступные для чтения, еще до его подключения. Этот документ не рассматривает соответствующие изменения API.

Клиент принимает SYN-ACK.

При получении пакета SYN-ACK клиенту **следует** выполнить указанные ниже действия.

1. Если SYN-ACK включает опцию Fast Open, опцию MSS или обе, обновляется соответствующее значение cookie и данные MSS в кэше cookie.
2. Передается пакет ACK с установкой в номере подтверждения RCV.NXT и включением данных после SND.UNA, если они доступны.
3. Переход в состояние ESTABLISHED.

Отметим, что здесь не возникает «наказания» за задержку, если отправитель не подтверждает данные в исходном пакете SYN. Клиенту **следует** повторно передать неподтвержденные данные в первом пакете ACK п. 2. Обмен данными начнется после согласования, как при обычном соединении TCP.

Если у клиента истекло время ожидания и повторно переданы лишь обычные пакеты SYN, он может эвристически определить пути, где преднамеренно отбрасываются пакеты SYN с опцией Fast Open или данными. Если SYN-ACK подтверждает лишь начальную последовательность и не включает опцию Fast Open cookie, это предположительно вызывается повторным (обычным) пакетом SYN, а исходный пакет SYN или соответствующий SYN-ACK потерян.

Сервер принимает ACK.

При получении пакета ACK, подтверждающего последовательность SYN, сервер декрементирует PendingFastOpenRequests и переходит в состояние ESTABLISHED. Какой-либо особой обработки не требуется.

5. Вопросы безопасности

Fast Open Cookie не позволяет атакующему просто организовать лавину поддельных пакетов SYN с данными, чтобы исчерпать ресурсы сервера или организовать атаку с усиленным отражением на другой хост. Сервер может защищаться от атак подставными пакетами SYN с непригодными cookie, используя имеющиеся методы [RFC4987]. Следует отметить, что генерация поддельных cookie «ничего не стоит», проверка пригодности cookie, присущая любой схеме аутентификации, может требовать гораздо больших ресурсов, нежели обработка обычных пакетов SYN. Ниже подробно описаны новые уязвимости TFO и меры противодействия.

5.1. Атаки SYN Flood с действительными cookie

Атакующий может получать cookie от скомпрометрированных (взломанных) хостов и лавинно рассылать фиктивные пакеты SYN с данными и «действительными» cookie (с взломанных хостов или иных удобных точек). Как и обычное согласование TCP, механизм TFO уязвим для таких атак. Однако ущерб здесь может быть существенно больше. Помимо временного нарушения работы портов сервиса возможно истощение ресурсов CPU и памяти на сервере. Такая атака будет отражаться в системных журналах сервера как DoS[1]-атака на прикладном уровне со стороны бот-сетей, вызывая средства защиты и сигналы тревоги.

Для защиты сервера важно ограничить максимальное число ожидающих запросов на соединения TFO, т. е. PendingFastOpenRequests (параграф 4.2). При достижении предела сервер временно отключает TFO, как указано в параграфе 4.1.2. Обработка Cookie на сервере и последующие запросы TFO будут обрабатываться как обычные запросы на соединение (данные будут отбрасываться, а подтверждаться будут лишь SYN). Это позволяет использовать обычные методы защиты от атак SYN flood [RFC4987], подобные SYN cookie для сохранения работоспособности сервера. Основным влиянием атак SYN flood на стандартный стек TCP является не сам лавинный трафик, потребляющий ресурсы на обработку TCP и память хоста, а обманные пакеты SYN, заполняющие небольшие очереди прослушивающих сокетов.

Однако с TFO атаки SYN flood могут напрямую препятствовать работе, если не задано ограничения в стеке. Пакеты RST от обманных хостов будут скорее усиливать, нежели препятствовать лавине пакетов SYN по сравнению с обычным (без TFO) случаем, поскольку атакующий может заполнить много пакетов SYN данными и это потребует больше ресурсов для обработки. По этой причине серверу TFO нужно отслеживать сбрасываемые соединения в состоянии SYN-RCVD в дополнение к обычному ограничению максимального размера очереди. Реализации могут

[1]Denial of service — отказ в обслуживании.

комбинировать эти методы, например, учитывая запросы на соединения, которые были только что сброшены по PendingFastOpenRequests, пока не истечет время ожидания.

Ограничение максимального числа ожидающих запросов TFO позволяет атакующему легко переполнить очередь, вызывая отключение TFO. Однако такой результат вряд ли будут интересен для атакующего, поскольку сохранится сервис без TFO и реальных нарушений работы практически не будет заметно.

5.1.1. Атаки из-за трансляторов NAT

Атакующий, расположенный за транслятором NAT, может легко получить действительные cookie для организации упомянутой выше атаки и нанести вред другим клиентам, использующим тот же путь. В [BRISCOE12] предложено серверам расширить создание cookie путем включения временных меток TCP - GetCookie(IP_Address, Timestamp) - и реализовать это путем шифрования конкатенации двух значения для создания cookie. Клиент сохраняет cookie и временную метку для возврата обоих значений в пакетах SYN. Затем сервер реализует IsCookieValid(IP_Address, Timestamp, Cookie), шифруя IP и временную метку для сравнения со значением cookie.

Это позволяет серверу создавать разные cookie для клиентов, использующих один адрес IP, что дает возможность отбрасывать неправомерно используемые злоумышленниками cookie. Однако злоумышленник может просто повторять атаки с новыми cookie. В конечном итоге серверу приходится подавлять все запросы с IP-адреса, как при текущем подходе. Кроме того, этот подход требует изменения [RFC1323] (отменен [RFC7323]) для передачи ненулевых zero Timestamp Echo Reply в пакетах SYN, что может вызывать проблемы с межсетевыми экранами. В результате преимущества не превышают недостатков.

5.2. Атаки с усиленным отражением на сторонние хосты

Ограничение PendingFastOpenRequests системным пределом можно выполнить без Fast Open cookie и это защитит ресурсы сервера, а также ограничит ущерб, который атакующий может нанести с помощью атак с усиленным сервером отражением. Однако остается уязвимость к атакам с усиленными отражениями от большого числа серверов. Атакующий может легко нанести ущерб, заставив многочисленные серверы отвечать пакетами данных по выбранному им IP-адресу жертвы.

При использовании Fast Open cookie атакующему потребуется сначала украсть действительное значение cookie у своей жертвы. Это вероятно потребует от атакующего сначала взломать хост или сеть жертвы, но в некоторых случаях это можно сделать сравнительно легко. В этом случае у злоумышленника уже не будет интереса в организации атаки на хост, который уже взломан, однако может представлять интерес нарушение работы сети жертвы. Поскольку украденное значение cookie пригодно лишь для одного сервера, атакующему потребуется украсть действительные cookie для множества серверов и использовать их до истечения срока действия, чтобы нанести существенный вред без срабатывания защиты.

Можно утверждать, что взломавший сеть или хост злоумышленник может организовать похожую, но более простую атаку путем простой вставки битов. Уровень ущерба будет идентичен, но связанная с TFO атака позволяет злоумышленнику сохранить анонимность и маскирует источник атаки. Например, атакующий может с помощью DHCP получить cookie, когда он (или взломанный им хост) владеет определенным IP-адресом, выполняя обычный механизм Fast Open для серверов, поддерживающих TFO и собирая действительные cookie. Затем атакующий активно или пассивно освобождает свой IP-адрес и когда этот адрес будет назначен сервером DHCP другому хосту (жертве), атакующий может организовать лавину обманных запросов Fast Open с действительными cookie к соответствующим серверам. Поскольку значения cookie действительны, серверы будут воспринимать запросы и отвечать пакетами SYN-ACK с данными по адресу жерты. Таким образом, злоумышленник может организовать атаку с усиленным отражением на другой хост.

Лучшей защитой для сервера будет не отвечать пакетами с данными до завершения процедуры согласования и это полностью исключает риск атак с усиленным отражением. Однако возможная экономия времени за счет использования TFO в этом случае будет снижаться.

6. Применимость TFO

Этот раздел предназначен в помощь приложениям, рассматривающим вопрос применения TFO, в оценке преимуществ и недостатков TFO на примере клиентских и серверных приложений Web. Приложениями в данном случае считаются процессы, напрямую записывающие данные в сокет, например, процесс JavaScript, передающий данные серверу. Предлагаемое изменение API сокета описано в Приложении A.

6.1. Дубликаты данных в SYN

При использовании TFO возможна неоднократная доставка первых данных, записанных в сокет, приложению на удаленном хосте (параграф 2.1). Это относится лишь к данным в пакетах SYN, но не к последующему обмену.

Опытным путем в [JIDKT07] показано, что дублирование пакетов в сети Tier-1 возникает редко. Поскольку повторы возможны лишь при дублировании пакетов SYN с данными, а дубликат приходит уже после того, как получатель сбросит исходное состояние SYN для соединения, влияние повторов представляется необычным на практике. Тем не менее, клиенту, который не может неоднократно обрабатывать одни и те же данные SYN, **недопустимо** включать TFO для отправки данных в SYN. Аналогично, серверу, не способному неоднократно воспринимать одни и те же данные в SYN, **недопустимо** включать TFO для приема данных в SYN. Для оценки вероятности получения дубликатов SYN и SYN-ACK с данными в сетях, отличных от Tier 1, нужны дополнительные исследования.

6.2. Возможное влияние на производительность

Механизм TFO разработан для приложений, чувствительных к задержкам на этапе организации соединений TCP. Для использования преимуществ TFO первый блок данных приложения (например, запрос HTTP) должен быть не больше максимального сегмента TCP (за вычетом размера опций в SYN). В противном случае удаленный сервер сможет обработать данные приложения лишь по завершении начального согласования, что снижает преимущества TFO.

По-возможности приложениям **следует** повторно использовать соединение для получения преимуществ встроенного контроля перегрузок TCP и снижения времени организации соединения. Приложение, использующее слишком много краткосрочных соединений, будет негативно влиять на стабильность сети, поскольку эти соединения зачастую завершаются до того, как начнет работать контроль перегрузок TCP.

6.3. Пример клиентов и серверов Web

6.3.1. Воспроизведение запроса HTTP

Хотя TFO разрабатывался для Web-приложений, браузерам не следует применять TFO для отправки запросов SYN, если они восприимчивы к повторам (replay). Одним из примеров являются запросы POST без защиты транзакций на прикладном уровне (например, с помощью уникального идентификатора в заголовке запроса).

С другой стороны, механизм TFO особенно полезен для запросов GET, поскольку воспроизведение (replay) GET может происходить в чередующихся (striped) соединениях TCP - после того, как сервер получит запрос HTTP, но до того, как ACK для запросов придут в браузер, в браузере может завершиться время ожидания и он повторит тот же запрос в другом (возможно новом) соединении TCP. Это отличается от воспроизведения TFO лишь тем, что повторное использование инициируется браузером, а не стеком TCP.

6.3.2. HTTP с использованием TLS (HTTPS)

Для TLS по протоколу TCP безопасно и полезно включать TLS client_hello в пакет SYN для экономии одного интервала RTT при согласовании TLS. Здесь нет опасений в части нарушения идемпотентности и, в частности, это может применяться вместе описанными выше соединениями.

6.3.3. Сравнение с постоянными соединениями HTTP

Полезен ли механизм TFO для широкого развертывания постоянных соединений HTTP? Кратким ответом будет «да». Исследования ([RCCJR11] [AERG11]) показали, что среднее число транзакций на соединение составляет 2 - 3 на основе масштабных измерений для серверов и клиентов. В этих исследованиях серверы и клиенты сохраняли бездействующие соединения на несколько минут (пока человек думает).

Сохранение бездействующих соединений дольше ведет к риску снижения производительности. В [HNESSK10] и [MQXMZ11] показано, что большинство домашних маршрутизаторов и ISP не соблюдают 124-минутный тайм-аут, заданный в [RFC5382]. В [MQXMZ11] отмечено, что 35% мобильных ISP просто обрывают соединения, бездействующие в течение 30 минут. Конечные хосты, не подозревая о тайм-аутах промежуточных устройств, используют продолжительное время ожидания для TCP для таких бездействующих соединений.

Для обхода этой проблемы некоторые приложения передают частые пакеты TCP keep-alive. Однако это ведет к истощению батарей на мобильных устройствах [MQXMZ11]. Фактически питание стало важным аспектом для современных устройств LTE (Long Term Evolution) и мобильные браузеры закрывают соединения HTTP в течение нескольких секунд или даже быстрее [SOUDERS11].

В [RCCJR11] исследована производительность браузера Chrome [Chrome] на основе статистики за 28 дней. Chrome сохраняет бездействующие постоянные соединения HTTP от 5 до 10 минут. Однако среднее число транзакций на соединение составляет лишь 3,3, а на TCP 3WHS приходится до 25% задержки сетевых транзакций HTTP. По оценке авторов TFO сокращает время загрузки страниц популярных сайтов на 10% - 40%.

6.3.4. Балансировщики нагрузки и серверные фермы

Серверам, расположенным за балансировщиками нагрузки и воспринимающим запросы по одному адресу IP, следует использовать общий ключ, обеспечивающий создание идентичных Fast Open cookie для определенного IP-адреса клиента. В противном случае клиент может получать в соединении разные cookie и попытки использовать Fast Open будут приводить к обычному согласованию 3WHS.

7. Открытые области для экспериментов

Здесь отмечены некоторые вопросы, которые требуют экспериментов в Internet и различных сетевых сценариях. Эти эксперименты должны помочь в оценке преимуществ и рисков Fast Open и связанных с этим протоколов.

7.1. Влияние промежуточных устройств и NAT на производительность

В [MAF04] отмечено, что некоторые промежуточные устройства и конечные хосты могут отбрасывать пакеты с неизвестными опциями TCP. Исследования ([LANGLEY06] [HNRGHT11]) показали, что 6% проверенных путей в Internet отбрасывали пакеты SYN с данными или неизвестными опциями TCP. TFO при возникновении такой проблемы возвращается к обычному согласованию TCP и повторяет SYN без данных и опций cookie по истечении начального тайм-аута SYN. Кроме того, реализациям рекомендуется кэшировать такие события для предотвращения повторных отказов. Требуется дополнительное исследование влияния таких событий на производительность.

Еще один вопрос связан с потерей преимуществ TFO при работе через некоторые трансляторы адресов операторского класса (Carrier-Grade NAT или CGN). Обычно хосты за NAT используют общий адрес IP и будут получать от сервера одно значение cookie. Это не препятствует работе TFO. Но некоторые конфигурации CGN, где для каждого нового соединения TCP от данного физического хоста используется другой внешний адрес IP, препятствуют получению преимуществ TFO. Однако это не снижает производительности, как указано в параграфе 4.2.2. TCP Fast Open.

7.2. Влияние на контроль перегрузок

Хотя TFO не влияет напрямую на контроль перегрузок TCP, в некоторых случаях это возможно. При тайм-ауте SYN-ACK обычный TCP уменьшает начальное окно насыщения перед отправкой любый данных [RFC5681], однако с TFO сервер уже мог передать данные размером вплоть до начального окна.

Если сервер обслуживает в основном краткосрочные соединения, потеря SYN-ACK не оказывает такого влияния на уменьшение окна насыщения, как в обычном TCP. Это может дестабилизировать состояние сети. Соединения,

сталкивающиеся с потерями, могут повторять попытки и увеличивать перегрузку. Возможным решением является временное отключение Fast Open, если сервер сталкивается с потерей многих пакетов данных или SYN-ACK в процессе согласования соединений. Здесь будут полезны дополнительные эксперименты.

7.3. Fast Open без cookie

Механизм cookie снижает риск истощения ресурсов и атак с усилением отражения. Однако cookie не требуются, если сервер имеет защиту на прикладном уровне или устойчив к таким атакам. Например, Web-сервер, отвечающий лишь простыми откликами HTTP redirect, помещаемыми в SYN-ACK, может не заботиться об истощении ресурсов.

Для таких приложений сервер может выбрать создание тривиальных и даже пустых cookie с целью повышения производительности за счет отказа от создания и проверки cookie. Если сервер обнаружит DoS-атаку с помощью других механизмов защиты, он может перейти на обычный режим Fast Open для прослушиваемых сокетов.

8. Смежные работы

8.1. T/TCP

Расширения TCP для транзакций (TCP Extensions for Transactions) [RFC1644] были (среди прочего) попыткой обойти 3WHS, поэтому цель их совпадает с целью TFO. Основные усилия были сосредоточены на борьбе со старыми и дублированными пакетами SYN, но не было уделено внимания уязвимостям, связанным с обходом 3WHS [PHRACK98].

Как отмечено выше, здесь применяется практический подход, сосредоточенный на безопасности TFO, допуская старые и продублированные пакеты SYN с данными после признания того, что полное соответствие семантике TCP скорей всего невозможно. Предлагается считать такой подход разумным компромиссом, поскольку это существенно упрощает TFO и делает механизм привлекательным для разработчиков и пользователей TCP.

8.2. Общая защита от атак SYN Flood

В [RFC4987] изучено ослабление обычных атак SYN flood (пакетами SYN без данных). Но не SYN cookie без учета состояния, ни SYN Cache с поддержкой состояний не защищают данные в пакетах SYN.

Более эффективной защитой может быть простое отключение TFO, когда на хосте предполагается атака SYN flood (например, заполнение SYN backlog). После отключения TFO применяется обычная защита от атак SYN flood. В разделе 5. Вопросы безопасности этот вопрос рассмотрен подробно.

8.3. Самостоятельно организуемые приложением соединения

Некоторые Web-браузеры поддерживают историю доменов для часто посещаемых страниц Web. Затем эти браузеры заранее открывают соединения TCP с такими доменами до того, как пользователь введет запрос к серверу [BELSHE11]. Хотя этот метод также снижает задержку, он расходует ресурсы сервера и сети, инициируя и поддерживая бездействующие соединения.

8.4. Fast Open Cookie в пакетах FIN

Другим предложением является запрос TFO в пакетах FIN, поскольку отбрасывание FIN несовместимыми промежуточными устройствами не влияет на задержку. Однако пути, блокирующие SYN cookie, с большей вероятностью могут отбросить последующий пакет SYN с данными, а многие приложения закрывают соединение с помощью RST, а не FIN.

Хотя cookie в пакетах FIN могут не повысить отказоустойчивость, это дает клиентам, использующим одно соединение, преимущество в задержке по сравнению с клиентами, создающими много параллельных соединений. Если эксперименты с TFO покажут рост совместног использования соединений, cookie в FIN могут быть полезны.

8.5. Транзакция TCPCT

TCPCT[1] [RFC6013] устраняет состояние сервера при начальном согласовании и защищает от DoS-атак с подменой. Подобно TFO, TCPCT позволяет включать данные в пакеты SYN и SYN-ACK. Но сервер в процессе начального согласования не может передать данных больше MSS (а не начального окна, как в TFO). Поэтому задержка для приложений (например, Web) может быть больше, чем при использовании TFO.

9. Взаимодействие с IANA

Агентство IANA выделило значение 34 в реестре TCP Option Kind Numbers (см. параграф 4.1.1). Новая опция TCP имеет переменный размер, а в поле Meaning реестра TCP Option Kind Numbers указано TCP Fast Open Cookie. Имеющимся и новым реализациям **следует** использовать номер опции 34. Реализациям, использующим номер 254 [RFC6994] с magic number 0xF989 (16 битов), выделенный в реестре TCP Experimental Option Experiment Identifiers (TCP ExIDs), **следует** перейти по умолчанию к новому номеру (34).

10. Литература

10.1. Нормативные документы

[RFC793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981, <http://www.rfc-editor.org/info/rfc793>.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989, <http://www.rfc-editor.org/info/rfc1122>.

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

[1]TCP Cookie Transaction.

[RFC3390] Allman, M., Floyd, S., and C. Partridge, "Increasing TCP's Initial Window", RFC 3390, October 2002, <http://www.rfc-editor.org/info/rfc3390>.

[RFC5382] Guha, S., Ed., Biswas, K., Ford, B., Sivakumar, S., and P. Srisuresh, "NAT Behavioral Requirements for TCP", BCP 142, RFC 5382, October 2008, <http://www.rfc-editor.org/info/rfc5382>.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, September 2009, <http://www.rfc-editor.org/info/rfc5681>.

[RFC6994] Touch, J., "Shared Use of Experimental TCP Options", RFC 6994, August 2013, <http://www.rfc-editor.org/info/rfc6994>.

10.2. Дополнительная литература

[AERG11] Al-Fares, M., Elmeleegy, K., Reed, B., and I. Gashinsky, "Overclocking the Yahoo! CDN for Faster Web Page Loads", in Proceedings of Internet Measurement Conference, November 2011.

[BELSHE11] Belshe, M., "The Era of Browser Preconnect", February 2011, <http://www.belshe.com/2011/02/10/the-era-of-browser-preconnect/>.

[BRISCOE12] Briscoe, B., "Some ideas building on draft-ietf-tcpm-fastopen-01", message to the tcpm mailing list, July 2012, <http://www.ietf.org/mail-archive/web/tcpm/current/msg07192.html>.

[Chrome] Google Chrome, <https://www.google.com/intl/en-US/chrome/browser/>.

[HNESSK10] Haetoenen, S., Nyrhinen, A., Eggert, L., Strowes, S., Sarolahti, P., and M. Kojo, "An Experimental Study of Home Gateway Characteristics", in Proceedings of Internet Measurement Conference, October 2010.

[HNRGHT11] Honda, M., Nishida, Y., Raiciu, C., Greenhalgh, A., Handley, M., and H. Tokuda, "Is it Still Possible to Extend TCP?", in Proceedings of Internet Measurement Conference, November 2011.

[JIDKT07] Jaiswal, S., Iannaccone, G., Diot, C., Kurose, J., and D. Towsley, "Measurement and Classification of Out-of-Sequence Packets in a Tier-1 IP Backbone" IEEE/ACM Transactions on Networking (TON), Volume 15, Issue 1, pp 54-66.

[LANGLEY06] Langley, A., "Probing the viability of TCP extensions", <http://www.imperialviolet.org/binary/ecntest.pdf>.

[MAF04] Medina, A., Allman, M., and S. Floyd, "Measuring Interactions Between Transport Protocols and Middleboxes", in Proceedings of Internet Measurement Conference, October 2004.

[MQXMZ11] Wang, Z., Qian, Z., Xu, Q., Mao, Z., and M. Zhang, "An Untold Story of Middleboxes in Cellular Networks", in Proceedings of SIGCOMM, August 2011.

[PHRACK98] "T/TCP vulnerabilities", Phrack Magazine, Volume 8, Issue 53, Article 6, July 8, 1998, <http://www.phrack.com/issues.html?issue=53&id=6>.

[RCCJR11] Radhakrishnan, S., Cheng, Y., Chu, J., Jain, A., and B. Raghavan, "TCP Fast Open", in Proceedings of the 7th ACM CoNEXT Conference, December 2011.

[RFC1323] Jacobson, V., Braden, R., and D. Borman, "TCP Extensions for High Performance", RFC 1323, May 1992, <http://www.rfc-editor.org/info/rfc1323>.

[RFC1644] Braden, R., "T/TCP -- TCP Extensions for Transactions Functional Specification", RFC 1644, July 1994, <http://www.rfc-editor.org/info/rfc1644>.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460[1], December 1998, <http://www.rfc-editor.org/info/rfc2460>.

[RFC4987] Eddy, W., "TCP SYN Flooding Attacks and Common Mitigations", RFC 4987, August 2007, <http://www.rfc-editor.org/info/rfc4987>.

[RFC6013] Simpson, W., "TCP Cookie Transactions (TCPCT)", RFC 6013, January 2011, <http://www.rfc-editor.org/info/rfc6013>.

[RFC6247] Eggert, L., "Moving the Undeployed TCP Extensions RFC 1072, RFC 1106, RFC 1110, RFC 1145, RFC 1146, RFC 1379, RFC 1644, and RFC 1693 to Historic Status", RFC 6247, May 2011, <http://www.rfc-editor.org/info/rfc6247>.

[RFC7323] Borman, D., Braden, B., Jacobson, V., and R. Scheffenegger, Ed., "TCP Extensions for High Performance", RFC 7323, September 2014, <http://www.rfc-editor.org/info/rfc7323>.

[SOUDERS11] Souders, S., "Making A Mobile Connection", <http://www.stevesouders.com/blog/2011/09/21/making-a-mobile-connection/>.

Приложение A. Пример изменения API сокета для поддержки TFO

A.1. Активная сторона соединения

Для активной стороны создаваемого соединения изменяется или заменяется вызов connect(), который не принимает в качестве аргумента пользовательский буфер данных. Рекомендуется заменить вызов connect() для минимизации правок в API и снижения издержек приложений при развертывании.

Одним из решений, принятым в ядре Linux 3.7, служит добавление флага MSG_FASTOPEN для sendto() и sendmsg(). MSG_FASTOPEN помечает попытку передать данные в SYN как комбинацию connect() и sendto(), за счет неявного вызова операции connect(). Блокировка сохраняется до завершения согласования и буферизации данных.

[1] Заменен RFC 8200. *Прим. перев.*

Для неблокируемого сокета возвращается число байтов, буферизованных и отправленных в пакете SYN. Если значение cookie не доступно локально, возвращается -1 с кодом ошибки EINPROGRESS и автоматически передается SYN с запросом TFO cookie. Вызывающему нужно снова записать данные при подключении сокета. При ошибке возвращается тот же код, что и в connect() при отказе в процессе согласования.

Реализация может отказаться от изменения вызова sendmsg(), поскольку TFO связан лишь с протоколом TCP. Решением может служить добавление новой опции сокета TCP_FASTOPEN. Когда эта опция установлена до операции connect(), вызов sendmsg() или sendto() будет выполнять Fast Open подобно описанному выше флагу MSG_FASTOPEN. Однако такой подход требует дополнительного системного вызова setsockopt().

A.2. Пассивная сторона соединения

Изменения на пассивной стороне проще и приложению нужно лишь разрешить прием запросов Fast Open с помощью новой опции TCP_FASTOPEN при вызове setsockopt() до вызова listen().

Опция включает Fast Open на прослушиваемом сокете. Значение опции задает порог PendingFastOpenRequests, т. е. макимальный размер ожидающих SYN с данными. После включения реализация TCP будет отвечать на запрос пакетом с TFO cookie. Традиционно accept() возвращает управление лишь после подключения сокета, но для соединений Fast Open accept() возвращает управление при получении пакета SYN с действительным Fast Open cookie и данными и доступности данных для чтения с помощью, например, recvmsg(), read().

Благодарности

Спасибо Bob Briscoe, Michael Scharf, Gorry Fairhurst, Rick Jones, Roberto Peon, William Chan, Adam Langley, Neal Cardwell, Eric Dumazet, Matt Mathis за их отклики. Особая благодарность Barath Raghavan за вклад по вопросам безопасности Fast Open и многократное вычитывание документа.

Адреса авторов

Yuchung Cheng

Google, Inc.

1600 Amphitheatre Parkway

Mountain View, CA 94043

United States

EMail: ycheng@google.com

Jerry Chu

Google, Inc.

1600 Amphitheatre Parkway

Mountain View, CA 94043

United States

EMail: hkchu@google.com

Sivasankar Radhakrishnan

Department of Computer Science and Engineering

University of California, San Diego

9500 Gilman Drive

La Jolla, CA 92093-0404

United States

EMail: sivasankar@cs.ucsd.edu

Arvind Jain

Google, Inc.

1600 Amphitheatre Parkway

Mountain View, CA 94043

United States

EMail: arvind@google.com

Перевод на русский язык
Николай Малых

nmalykh@protokols.ru