Network Working Group
Request for Comments: 4656
Category: Standards Track

S. Shalunov
B. Teitelbaum
A. Karp
J. Boote
M. Zekauskas
Internet2

September 2006

A One-way Active Measurement Protocol (OWAMP)

Протокол одностороннего активного измерения (OWAMP)

Статус документа

Этот документ задает проект стандартного протокола Internet для сообщества Internet и служит приглашением к дискуссии в целях развития протокола. Текущее состояние стандартизации и статус протокола можно узнать из документа «Internet Official Protocol Standards» (STD 1). Документ можно распространять без ограничений.

Авторские права

Аннотация

Протокол одностороннего активного измерения OWAMP служит для измерения в одном направлении таких характеристик, как задержка и потеря пакетов. Высокоточное измерение этих односторонних параметров производительности IP стало возможным благодаря доступности надежных источников точного времени (таких как GPS и CDMA). OWAMP обеспечивает совместимость измерений.

Оглавление

1. Введение

Рабочая группа IETF IPPM[1] определила метрику для односторонней задержки [RFC2679] и потери [RFC2680] пакетов на пути Internet. Хотя уже имеется несколько измерительных платформ ([SURVEYOR] [SURVEYOR-INET] [RIPE] [BRIX]) реализующих такую метрику, пока нет стандарта, позволяющего инициировать тестовые потоки или обмениваться пакетами для сбора отдельных параметров в совместимой манере.

Расширение сферы доступности недорогих систем глобального позиционирования (GPS[2]) и источников точного времени на базе CDMA, хосты все чаще применяют источники точного времени напрямую или через протокол NTP[3] от первичных (stratum 1) серверов точного времени. Путем стандартизации методов односторонних измерений IPPM можно надеяться создать среду, где параметры IPPM можно будет собирать через широкую сеть путей Internet, что не было возможно раньше. Одним из проявлений этого послужит широкое развертывание серверов OWAMP, которые сделают возможным одностороннее измерение задержки, а также времени кругового обхода с использованием основанных на ICMP инструментов, подобных ping.

Дополнительными мотивами разработки OWAMP стали сложность обнаружения и манипуляций с пакетами, безопасность, логическое разделение функций тестирования и управления, а также поддержка небольших пробных пакетов (сложность детектирования пакетов препятствует помехам со стороны промежуточных узлов сети).

Тестовый трафик OWAMP сложно обнаружить, поскольку он представляет собой обычный поток пакетов UDP с согласованными номарами портов, которые практически не имеют статических параметров (размер тоже согласуется). OWAMP также поддерживает шифрование, которое дополнительно скрывает трафик и препятствует незаметному изменению временных меток.

Функции безопасности включают дополнительную аутентификацию и/или шифрование управляющих и тестовых сообщений. Эти функции могут быть полезны для предотвращения несанкционированного доступа к результатам и MITM[4]-атак с попытками обеспечить специальную обработку тестовых потоков OWAMP или поменять созданные отправителем временные метки для фальсификации результов.

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 интерпретируются в соответствии с [RFC2119].

1.1. Связи между протоколами тестирования и управления

OWAMP фактически состоит из двух связанных протоколов - OWAMP-Control и OWAMP-Test. OWAMP-Control служит для инициирования, запуска и остановки тестовых сессий, а также для сбора их результатов, а OWAMP-Test применяется для обмена тестовыми пакетами между измерительными узлами.

Хотя OWAMP-Test можно применять с другим (не OWAMP-Control) протоколом управления, авторы осознанно включили оба протокола в один RFC для стимулирования реализации и развертывания OWAMP-Control в качестве общего протокола управления разделителями для односторонних измерений. Наличие полного и открытого решения для односторонних измерений, которое легко реализовать и развернуть, имеет решающее значение для будущего в плане внутридоменных односторонних активных измерений, которые могут стать столь же привычным инструментом, как ping. Здесь не предлагается и не рекомендуется использовать OWAMP-Control как основу для универсального расширяемого решения в части управления мониторингом и измерениями.

OWAMP-Control разработан для поддержки согласования сессий одностороннего активного измерения и извлечения результатов простым способом. При инициировании сеанса происходит согласование адресов и номеров портов отправителя и получателя, времени начала сессии, ее продолжительности, размера тестовых пакетов, среднего интервала выборки Пуассона и некоторых атрибутов общего назначения [RFC 2330], включая размер пакета и поэтапное поведение (PHB[5]) [RFC2474], которые могут применяться для поддержки одностороннего измерения характеристик сети через сети с дифференцированным обслуживанием. В дополнение к этому OWAMP-Control поддерживает на уровне сессии шифрование и аутентификацию для тестового и управляющего трафика, серверы измерений, которые могут служить посредниками (proxy) для конечных точек тестового потока, обмен значениями «затравок» (seed) для псевдослучайного процесса Пуассона, описывающего тестовый поток, генерируемый отправителем.

Предполагается, что OWAMP-Control сможет эффективно поддерживать односторонние измерения в разных средах с общедоступными измерителями на произвольных хостах для развертывания мониторинга в частных корпоративных сетях. Для интеграции с протоколом SNMP[6] или фирменными протоколами сетевого управления могут создаваться шлюзы.

1.2. Логическая модель

Имеется несколько логически разделенных ролей, которые обеспечивают гибкость применения.

Session-Sender
: Передающая сторона сессии OWAMP-Test.

Session-Receiver
: Принимающая сторона сессии OWAMP-Test.

Server
: Конечная система, поддерживающая одну или множество сессий OWAMP-Test и позволяющая настраивать состояния на уровне сессии в ее конечных точка, а также способная возвращать результаты сеанса тестирования.

[1] IP Performance Metrics - метрика производительности IP.
[2] Global positioning systems.
[3] Network Time Protocol - протокол сетевого времени.
[4] Man-in-the-middle - перехват и изменение пакетов с участием человека.
[5] Per-hop behavior.
[6] Simple Network Management Protocol - простой протокол сетевого управления.

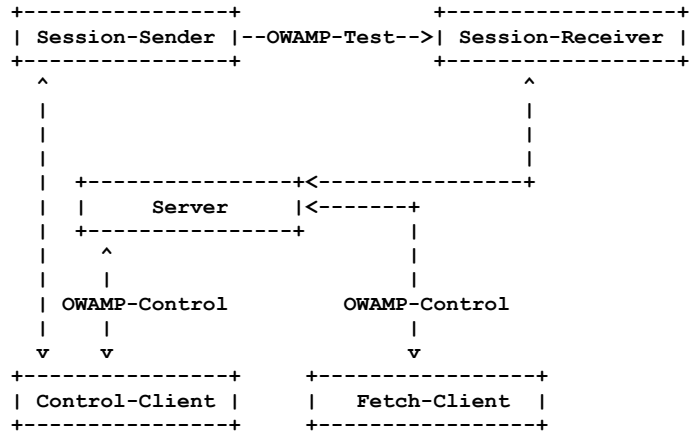
Control-Client

Конечная система, которая инициирует запрос сессий OWAMP-Test, запускает сессии и может инициировать их прерывание.

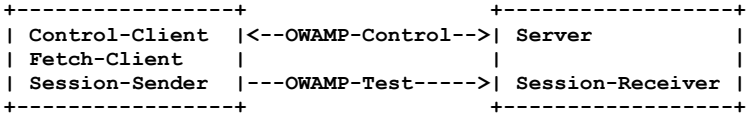
Fetch-Client

Конечная система, инициирующая запросы на извлечение результатов завершенных сессий OWAMP-Test.

Возможные связи между разными ролями показаны на рисунке. Связи без меток могут использовать не рассматриваемые в документе протоколы (включая фирменные).



Хост может выступать одновременно в нескольких ролях. Например, приведенная на рисунке схема может быть реализована на двух хостах, один из которых играет роли Control-Client, Fetch-Client и Session-Sender, а другой - Server и Session-Receiver, как показано ниже.



Поскольку многие пути в Internet включают сегменты IP на основе ATM, измерение задержки и потерь может испытывать влияние сегментации и сборки ATM (SAR[1]). Поэтому в OWAMP предусмотрено использование мелких тестовых пакетов, помещающихся в одну ячейку ATM (это возможно лишь в режиме бз аутентификации).

2. Обзор протокола

Как отмечено выше, OWAMP включает два связанных между собой протокола - OWAMP-Control и OWAMP-Test. Первый работает на основе TCP и служит для организации и управления сеансами измерения, а также для сбора результатов. Второй протокол работает на основе UDP и служит для передачи отдельных тестовых пакетов по проверяемому пути Internet.

Инициатор сеанса измерений организует соединение TCP через общеизвестный порт 861 и это соединение сохраняется на протяжении сессии OWAMP-Test. Серверам OWAMP **следует** прослушивать указанный порт.

Сообщения OWAMP-Control передаются только до запуска сессий OWAMP-Test и после их завершения (исключением являются сообщения Stop-Sessions).

Протоколы OWAMP-Control и OWAMP-Test поддерживают три режима работы - без аутентификации (unauthenticated), с аутентификацией (authenticated) и шифрованный (encrypted). Для двух последних режимов конечным точкам может потребоваться общий секрет.

Все многооктетные значения в этом документе представляются целыми числами без знака с сетевым порядкрм байтов, если явно не указано иное.

3. OWAMP-Control

Тип каждого сообщения OWAMP-Control можно определить после чтения первых 16 октетов. Размер сообщения OWAMP-Control можно рассчитать после считывания части сообщения фиксированного размера. Сообщений короче 16 октетов не бывает.

Реализациям **следует** исключать неиспользуемые состояния для предотвращения атак на службы или неограниченного расхода памяти на сервере. Например, если полное управляющее сообщение не было получено в течение некоторого числа минут после ожидаемого времени, соединение TCP, связанное с сессией OWAMP-Control, **следует** сбросить (drop). При отсутствии других соображений разумной верхней границей ожидания представляется значение 30 минут.

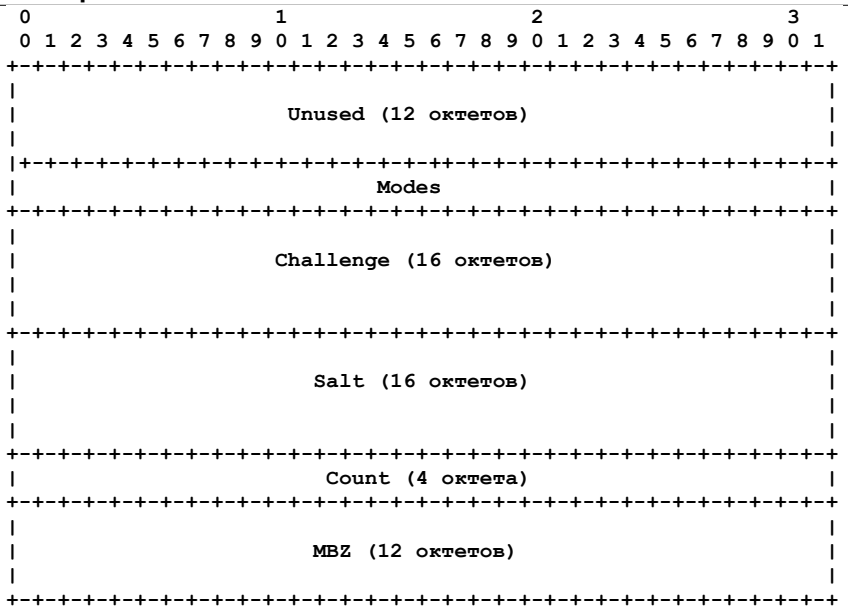
3.1. Организация соединения

Нужно организовать соединение с сервером (Server) до того, как Control-Client или Fetch-Client смогут подавать команды серверу.

Сначала клиент организует соединение TCP с общеизвестным портом 861 на сервере. Сервер будет отвечать приветственным сообщением, формат которого показан ниже.

Режим работы может принимать несколько значений: 1 - без аутентификации, 2 - с аутентификацией и 4 - с шифрованием. Значение поля Modes в сообщении сервера является результатом операции ИЛИ (OR) для всех значений, которые сервер готов поддерживать в этой сессии. Таким образом, используются три последних бита 32-битового поля Modes, а все предшествующие биты (29) **должны** иметь значение 0. Клиенты **должны** игнорировать первые 29 битов значения Modes (эти биты доступны для будущих расширений протокола).

[1]Segmentation and reassembly.



Поле Challenge содержит случайную последовательность октетов, создаваемую сервером, которая затем применяется клиентом для подтверждения владения общим секретом, как описано ниже

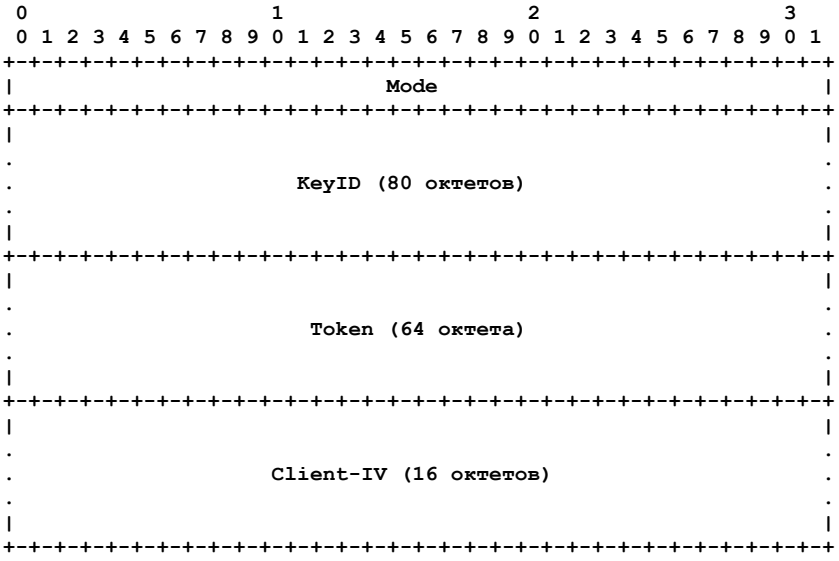
Параметры Salt и Count применяются при создании ключа из общего секрета, как описано ниже.

Значение Salt **должно** быть псевдослучайным (генерируется независимо от чего-либо иного в этом документе).

Значение Count **должно** быть степенью 2 и не менее 1024. Значение Count **следует** увеличивать по мере роста производительности расчетов.

Значение Modes = 0 указывает, что сервер не хочет взаимодействовать с клиентом и тот **может** незамедлительно разорвать соединение. Клиенту **следует** закрывать соединение при получении приветствия с Modes = 0. Клиент **может** закрыть соединение, если желаемый для него режим не доступен.

В иных случаях клиент **дожен** вернуть показанное ниже сообщение Set-Up-Response.



Поле Mode здесь указывает выбранный клиентом режим для данной сессии OWAMP-Control. Этот режим будет также применяться для всех сессий OWAMP-Test, начатых под управлением этой сессии OWAMP-Control. В поле Mode **должен** быть установлен один из трех последних битов или сброшены все биты. Если один из трех последних битов установлен, этот бит **должен** указывать один из поддерживаемых сервером режимов (т. е. этот бит должен быть установлен в полученном ранее приветствии от сервера). Первые 29 битов поля Mode **должны** быть сброшены (0). Сервер **должен** игнорировать значения этих битов. Если клиент установил Mode = 0, это говорит о том, что клиент не будет продолжать сессию. В этом случае клиенту и серверу **следует** закрыть соединение TCP, связанное с сессией OWAMP-Control.

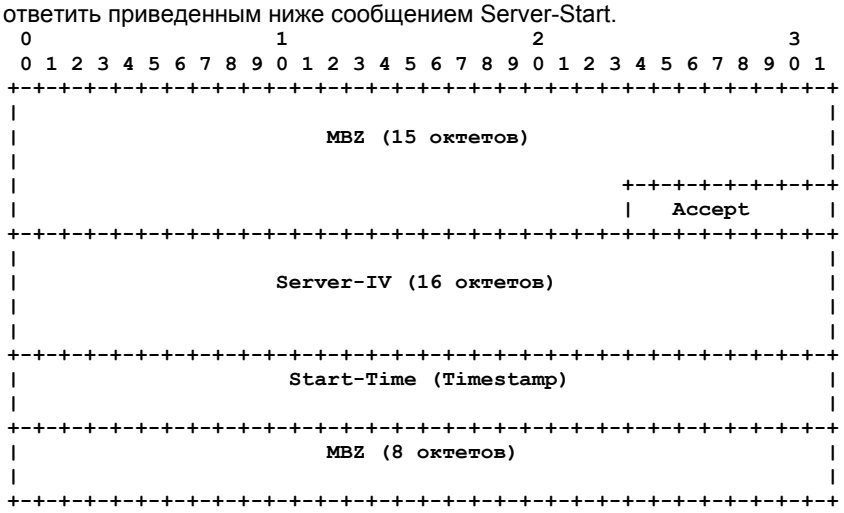
В режиме без аутентификации (1) поля KeyID, Token и Client-IV не используются. В остальных случаях KeyID содержит строку UTF-8 размером до 80 октетов (более короткие значения дополняются нулевыми октетами), которая указывает серверу общий секрет, выбранный клиентом для использования при проверке подлинности или шифровании, а Token содержит конкатенацию 16-октетного вызова (challenge), 16-октетного ключа AES Session-key для шифрования и 32-октетного ключа HMAC-SHA1 Session-key для аутентификации. Сам маркер шифруется с помощью алгоритма AES (Advanced Encryption Standard) [AES] в режиме CBC[1]. Шифрование **должно** выполняться с использованием вектора инициализации IV[2] = 0 и ключа, выведенного из общего секрета, связанного с KeyID (клиент и сервер используют общее отображение KeyIDs на значения секретов, сервер, готовый работать с множеством клиентов использует KeyID для выбора подходящего ключа, клиент обычно имеет свои ключи для каждого сервера; это похоже на использование паролей).

[1]Cipher Block Chaining - цепочка шифрованных блоков.
[2]Initialization Vector.

Общий секрет является парольной фразой и **должен** быть одной строкой (без символов перевода строки). Секретный ключ выводится из этой парольной фразы с использованием функции PBKDF2 (PKCS #5) [RFC2898]. PBKDF2 требует нескольких параметров - PRF является HMAC-SHA1 [RFC2104], a salt и count берутся из серверного сообщения.

Случайные значения AES Session-key, HMAC Session-key и Client-IV создаются клиентом. AES Session-key и HMAC Session-key **должны** генерироваться с достаточной энтропией, не снижающей уровень защиты базового шифра [RFC4086]. Значение Client-IV просто должно быть уникальным (т. е. **должны** быть разные в каждой сессии, использующей один секретный ключ). Простым способом решения этой задачи является генерация Client-IV с помощью криптографически стойкого источника псевдослучайных чисел. В таком случае вероятность повтора значения мала в течение 2^{64} сессий с одним секретным ключом.

Сервер **должен** ответить приведенным ниже сообщением Server-Start.



Поля MBZ[1] **должны** быть 0 и клиент **должен** игнорировать такие поля. Далее все поля MBZ имеют одинаковый смысл - передающая сторона **должна** заполнять все биты поля нулями, а принимающая сторона **должна** игнорировать эти поля (такие поля могут использоваться в будущих расширениях).

Случайное значение Server-IV генерируется сервером, а в режиме без аутентификации не применяется.

Поле Accept указывает отношение сервера к продолжению сессии. Accept=0 указывает, что сервер принял аутентификацию и готов продолжать транзакции. Отличное от 0 значение говорит о том, что сервер не принял аутентификацию или по иной причине не желает продолжать транзакции в этой сессии OWAMP-Control. Полный список доступных значений поля Accept приведен в параграфе 3.3.

Если передан негативный (не 0) отклик, сервер **может** (а клиенту **следует**) закрыть соединение после сообщения.

Start-Time содержит временную метку момента начала работы текущего экземпляра сервера (например, в многопользовательской операционной системе общего назначения это может быть время запуска серверного процесса). Если поле Accept отлично 0, все биты Start-Time **следует** заполнить 0. В режимах с аутентификацией и шифрованием поле Start-Time шифруется, как описано в параграфе 3.4, если Accept = 0 (режимы с аутентификацией и шифрованием не могут начать работу, пока не будет инициализировано управляющее соединение).

Формат поля Timestamp описан в параграфе 4.1.2. Экземпляру сервера **следует** сообщать одинаковое значение Start-Time каждому клиенту в каждой сессии.

Предыдущая транзакция считает соединение организованным.

3.2. Защита целостности (HMAC)

Аутентификация каждого сообщения (называется также командой) в OWAMP-Control обеспечивается добавлением HMAC. Протокол OWAMP применяет HMAC-SHA1 с отсечкой до 128 битов, т. е. поле HMAC имеет размер 16 октетов. Для HMAC нужен ключ. HMAC Session-key передается вместе с AES Session-key при организации соединения OWAMP-Control. Значение HMAC Session-key **следует** создавать независимо от AES Session-key (реализация **может** применять один механизм для генерации случайных битов обоих ключей). Каждое переданное в данном направлении значение HMAC учитывает все, что было передано в этом направлении после предыдущего HMAC (не включая его) и до начала нового кода HMAC. Таким путем после установки шифрования каждый бит соединения OWAMP-Control аутентифицируется кодом HMAC в точности один раз.

При использовании шифрования аутентификация выполняется до него, поэтому блоки HMAC шифруются вместе с остальным потоком. При расшифровке проверка HMAC выполняется для расширенных данных.

Значение HMAC **должно** проверяться как можно раньше для предотвращение использования и распространения поврежденных данных.

В открытом режиме поля HMAC не используются и имеют такую же семантику, как поля MBZ.

3.3. Значения поля Accept

Значения поля Accept используются протоколом OWAMP-Control для передачи сервером откликов на запросы клиента. Полный набор возможных значений приведен ниже.

0 OK (принято).

1 Отказ без указания причины (подходит все).

[1]MUST be zero - должно быть 0.

2 Внутренняя ошибка.

3 Некоторые аспекты запроса не поддерживаются.

4 Невозможно выполнить запрос по причине постоянного ограничения ресурсов.

5 Невозможно выполнить запрос по причине временного ограничения ресурсов.

Все прочие значения являются резервными. Отправитель сообщения **может** использовать значение 1 вместо всех отличных от 0 значений Accept. Отправителю **следует** корректное значение Accept, если оно отличается от 1. Получатель **должен** интерпретировать все отличающиеся от приведенных выше значения Accept как 1. Это обеспечивает доступность таких значений для будущих расширений.

3.4. Команды OWAMP-Control

В режиме с аутентификацией или шифрованием (они идентичны для OWAMP-Control и различаются лишь в OWAMP-Test) все последующие коммуникации шифруются с помощью AES Session-key (в режиме CBC) и аутентифицируются с использованием HMAC Session-key. Клиент шифрует все, что он передает в только что организованное соединение OWAMP-Control, применяя потоковое шифрование с Client-IV в качестве IV. Соответственно, сервер шифрует на свое стороне соединения с использованием Server-IV в качестве IV.

Сами векторы инициализации IV передаются в открытом виде. Шифрование начинается с блока, следующего непосредственно за блоком с IV. Два потока (от клиента к серверу и обратно) шифруются независимо со своими IV, но используют один ключ AES Session-key.

Для клиента доступны команды Request-Session, Start-Sessions, Stop-Sessions и Fetch-Session. Команда Stop-Sessions доступна клиенту и серверу (сервер также передает другие сообщения в ответ на полученные команды).

После передачи клиентом команды Start-Sessions и до отправки и получения (в любом порядке) команды Stop-Sessions клиент ведет активные измерения. О сервере говорят, что он ведет активные измерения с момента получения им команды Start-Sessions и до передачи и приема (в любом порядке) команды Stop-Sessions.

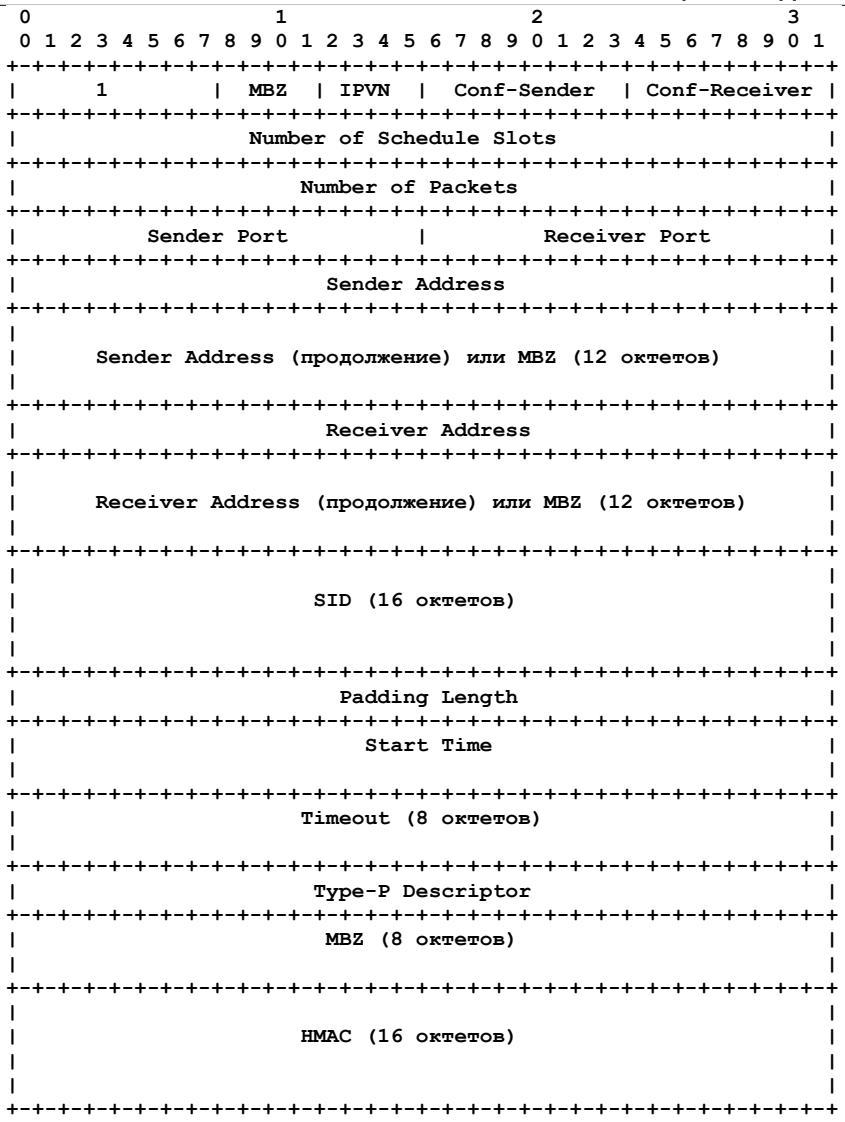
При выполнении активных измерений доступна лишь команда Stop-Sessions.

Более подробное описание команд приведено ниже.

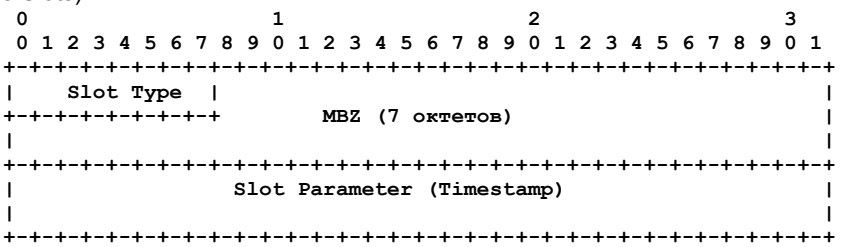
3.5. Организация тестовых сессий

Отдельные сеансы односторонних активных измерений организуются с использованием протокола запросов-откликов. Клиент OWAMP **может** (но не обязан) отправить множество сообщений Request-Session серверу OWAMP, который **должен** ответить на каждое сообщением Accept-Session. Это сообщение **может** отвергать запрос клиента.

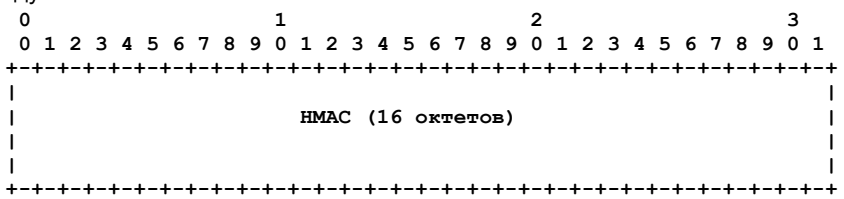
Формат сообщения Request-Session показан ниже.



Далее следует одно или несколько описаний временных интервалов расписания (число интервалов указывается полем Number of Schedule Slots)



Затем сразу же следует HMAC



Все эти сообщения образуют одну команду Request-Session.

На рисунке выше (начало страницы) первый октет со значением 1 указывает команду Request-Session.

IPVN указывает номера версии IP для отправителя и получателя. Для IP версии 4 за адресами IPv4 в Sender Address и Receiver Address следуют 12 неиспользуемых октетов. Эти октеты **должны** устанавливаться в 0 клиентом и **должны** игнорироваться сервером. В настоящее время осмысленными значениями IPVN являются 4 и 6.

В полях Conf-Sender и Conf-Receiver клиент **должен** указывать значение 0 или 1. Сервер **должен** интерпретировать отличные от 0 значения как 1. Если поле имеет значение 1, серверу предлагается настроить соответствующий агент (отправитель или получатель). В этом случае соответствующее значение Port серверу **следует** игнорировать. По крайней мере одно из значений Conf-Sender и Conf-Receiver **должно** быть 1 (при установке значения 1 для обоих серверу предлагается сессия между двумя хостами, которые он может настроить).

Поле Number of Schedule Slots, как отмечено выше, указывает число временных интервалов между двумя блоками HMAC. Оно применяется отправителем для определения времени передачи пакетов (см. следующий параграф).

Number of Packets указывает число пакетов активного измерения, передаваемых в течение данной сессии OWAMP-Test (отметим, что сессия может прервана раньше клиентом или сервером).

Если поле Conf-Sender не установлено (0), Sender Port задает порт UDP, из которого будут передаваться пакеты OWAMP-Test. Если поле Conf-Receiver не установлено (0), Receiver Port указывает порт UDP в который запрашивается передача пакетов OWAMP-Test.

Поля Sender Address и Receiver Address содержат, соответственно, адреса отправителя и получателя, являющихся конечными точками пути Internet, для которого запрашивается сессия тестирования OWAMP.

SID указывает идентификатор сессии. Он может применяться в последующих сессиях как аргумент команды Fetch-Session. Идентификатор имеет значение лишь при Conf-Receiver = 0. Таким образом, SID всегда генерируется принимающей стороной (информация о создании SID приведена в конце параграфа).

Поле Padding length указывает число октетов, добавляемых в конце обычного пакета OWAMP-Test (см. ниже).

Поле Start Time указывает время начала сессии (не раньше ввода команды Start-Sessions). Это временная метка в том же формате, что и метки OWAMP-Test.

Timeout (или порог потерь) задает интервал времени (в виде временной метки). Пакет, относящийся к тестовой сессии, организованной текущей командой Request-Session, будет считаться потерянным, если он не будет получен в течение Timeout после отправки.

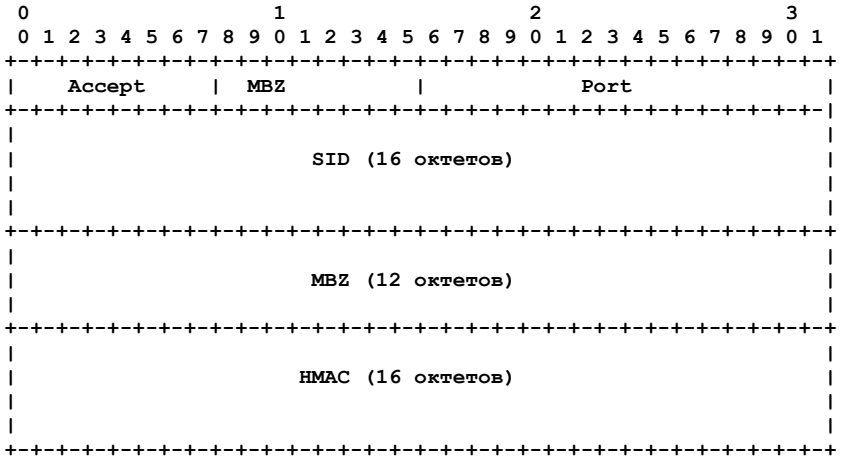
Type-P Descriptor покрывает только часть (очень большого) пространства Type-P. Если первые два бита Type-P Descriptor имеют значение 00, следующие 6 битов указывают запрашиваемое значение DSCP[1] для передаваемых пакетов OWAMP-Test, как определено в [RFC2474]. Если первые два бита Type-P имеют значение 01, следующие 16 битов указывают PHB ID[2], как определено в [RFC2836].

Значение, содержащее только 0, указывает принятое по умолчанию обслуживание best-effort.

Если поле Conf-Sender установлено (не 0), Type-P Descriptor служит для настройки отправителя на передачу пакетов в соответствии со значением этого поля. Если Conf-Sender = 0, Type-P Descriptor указывает, как будет настроен отправитель.

Если поле Conf-Sender установлено (не 0), а сервер не распознал Type-P Descriptor или не хочет устанавливать соответствующие атрибуты пакетов OWAMP-Test, ему **следует** отвергнуть запрос сессии. Если Conf-Sender =0, серверу **следует** воспринимать или отбрасывать сессию, не обращая внимания на значение Type-P Descriptor.

На каждый запрос Request-Session сервер OWAMP **должен** отвечать показанным ниже сообщением Accept-Session.



В этом сообщении поле Accept = 0 показывает, что сервер готов продолжать сессию, а все остальные значения отвергают запрос. Полный список значений поля Accept приведен в параграфе 3.3.

Если сервер отвергает сообщение Request-Session, ему **следует** не сохранять соединение TCP. Клиент **может** закрыть соединение, получив негативный отклик на сообщение Request-Session.

Смысл поля Port в отклике зависит от значений Conf-Sender и Conf-Receiver в вызвавшем отклик запросе. Если оба поля были установлены (не 0), поле Port не используется. Если было установлено лишь поле Conf-Sender, Port указывает номер порта, из которого ожидаются пакеты OWAMP-Test. Если было установлено только поле Conf-Receiver, Port указывает номер порта, в который передаются пакеты OWAMP-Test.

Если было установлено лишь поле Conf-Sender, поле SID в отклике не используется. В остальных случаях SID содержит уникальный идентификатор сессии, созданный сервером. Позднее он может применяться для сбора результатов данной сессии.

Значения SID **следует** создавать путем конкатенации 4 октетов адреса IPv4, относящегося к генерирующей идентификатор машине, 8-октетной временной метки и 4 октетов случайного значения. Для снижения вероятности конфликтов при наличии на генерирующей машине любых адресов IPv4 (за исключением loopback) один из таких адресов **следует** применять при генерации SID даже в случае коммуникаций на основе IPv6. Если адресов IPv4 нет совсем, **можно** использовать 4 последних октета адреса IPv6. Отметим, что значение SID всегда выбирается получателем. Если случайные значения не доступны, важно обеспечить непредсказуемость SID, поскольку значение идентификатора может быть использовано для контроля доступа.

[1]Differentiated Services Codepoint - код дифференцированного обслуживания.
[2]PHB Identification Code - код идентификации PHB.

3.6. Планирование передачи

Отправитель и получатель должны знать общее расписание передачи. Таким образом, при потере пакетов получатель знает, когда пакет должны были передать. Желательно сжимать общее расписание и сохранять возможность использования произвольного расписания для тестовых сессий. Во многих случаях расписание представляет собой повторяющиеся серии пакетов, выполняющие определенную проверку и повторяемые для сбора статистики.

Для реализации этого применяется планирование с заданным числом временных интервалов, каждый из которых имеет тип и параметр. Поддерживается два типа - псевдослучайные значения с экспоненциальным распределением (0) и фиксированные значения (1). Параметры указываются временными метками, задающими интервал времени. Для интервалов типа 0 (псевдослучайные значения с экспоненциальным распределением) этот интервал является средним значением (или 1/lambda, если плотность распределения имеет вид lambda*exp(-lambda*x) с положительными значениями x). Для интервалов типа 1 (фиксированное значение) параметр указывает задержку. Отправитель начинает выполнение расписания и исполняет инструкции временного интервала - для типа 0 время ожидания определяется экспоненциально распределенным значением со средним интервалом, заданным параметром, по истечении которого передается пакет (и обрабатывается следующий интервал), для слотов типа 1 время ожидания фиксировано. Расписание является циклическим и после выполнения всех интервалов отправитель возвращается к первому.

Отправитель и получатель должны быть способны воспроизводить расписание целиком (поэтому при потере пакета получатель все равно может связать с ним временную метку отправки). Воспроизведение интервалов типа 1 является тривиальной задачей. Для воспроизведения интервалов типа 0 нужна возможность воспроизводимой генерации псевдослучайных значений с экспоненциальным распределением. Способы решения этой задачи рассмотрены в разделе 5.

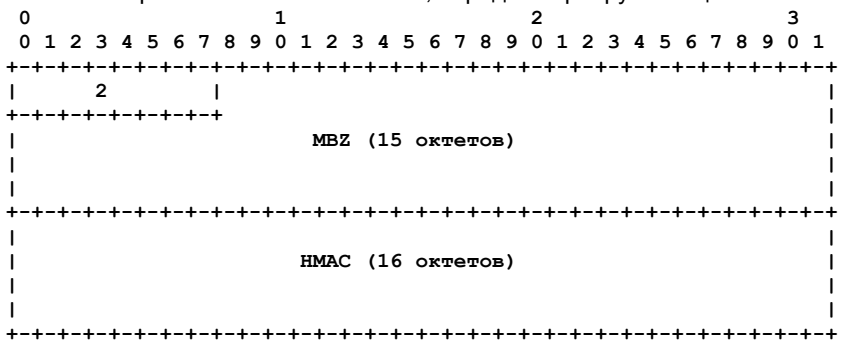
С использованием этого механизма можно легко задать базовые сценарии тестирования, примеры которых приведены ниже.

- Поток Пуассона - один интервал типа 0.
- Периодический поток - один интервал типа 1.
- Поток Пуассона из последовательных пар пакетов - два интервала типа 0 с отличным от 0 параметром и один интервал типа 1 с параметром 0.

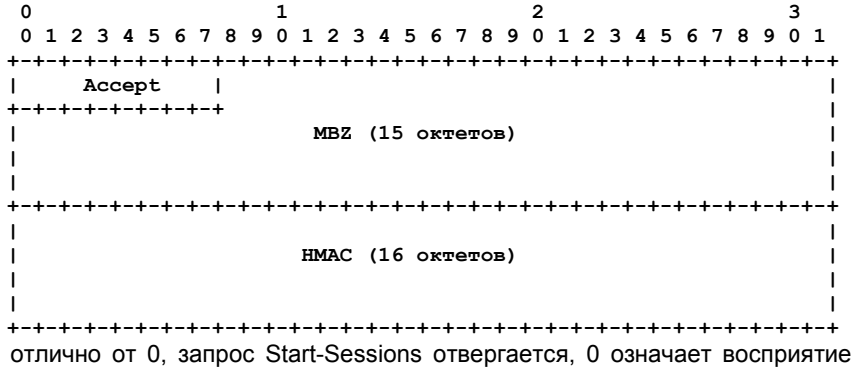
Кроме того, можно задать совершенно произвольное расписание (хоть это и неэффективно) путем создания множества тестовых пакетов, число которых совпадает с числом интервалов расписания. В этом случае полное расписание передается перед сессией OWAMP-Test.

3.7. Запуск тестовой сессии

Запросив одну или несколько тестовых сессий и получив положительные отклики Accept-Session, клиент OWAMP **может** начать выполнение запрошенного сеанса тестов, передав серверу сообщение Start-Sessions, показанное ниже.



Сервер **должен** ответить сообщением Start-Ack (его **следует** отправлять как можно скорей), показанным ниже.

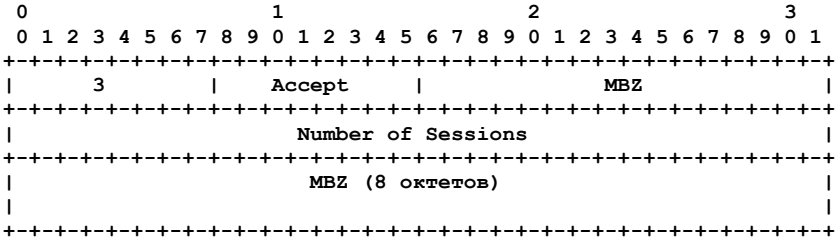


Если поле Accept отлично от 0, запрос Start-Sessions отвергается, 0 означает восприятие команды. Полный список возможных значений Accept дан в параграфе 3.3. Сервер **может**, а клиенту **следует** закрывать соединение отвергнутой сессии.

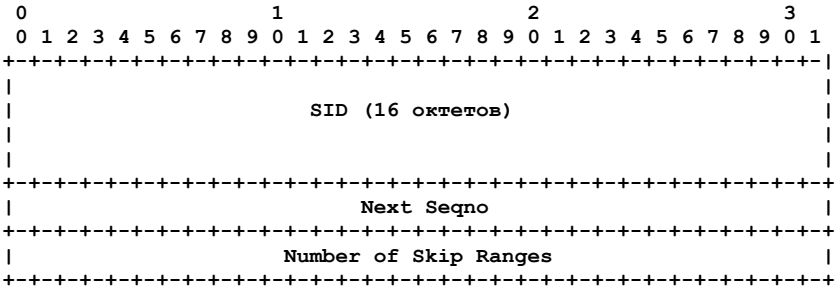
Серверу **следует** запускать все потоки OWAMP-Test сразу после отклика или заданного времени старта (что позднее). Если клиент представляет отправителя (Sender), ему **следует** щапускать свои потоки OWAMP-Test сразу после получения отклика Start-от сервера (если команда Start-Sessions воспринята) или заданного времени старта (что позднее). Поведение отправителя OWAMP-Test более подробно описано ниже.

3.8. Stop-Sessions

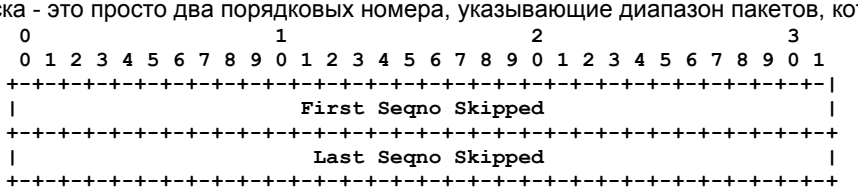
Сообщения Stop-Sessions может передавать Control-Client или сервер. Форма сообщения показан ниже.



Сразу за этим могут (не обязательно) следовать описания сессий (их число задано полем Number of Sessions). Описание сессии служит для указания пакетов, которые были действительно переданы отправляющим процессом (не пропущены). Заголовок описания сессии показан ниже.

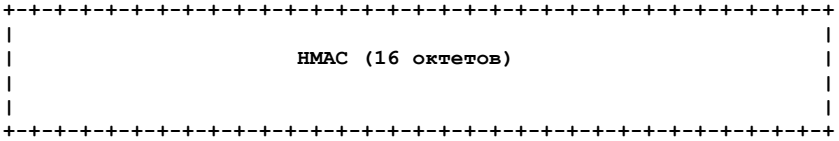


Сразу за этим могут (не обязательно) следовать описания Skip Range, как указано полем Number of Skip Ranges. Диапазоны пропуска - это просто два порядковых номера, указывающие диапазон пакетов, которые не были переданы.



Значения Skip Range **должны** быть упорядочены. Последний (возможно, неполный) блок данных (16 октетов) **должен** быть дополнен нулями при необходимости. Это гарантирует начало следующего описания сессии на границе блока.

В заключение добавляется один блок (16 октетов) HMAC, завершающий сообщение Stop-Sessions.



Все эти записи образуют одно логическое сообщение - команду Stop-Sessions.

Первый октет со значением 3 в приведенном выше формате указывает команду Stop-Sessions.

Отличное от 0 значение Accept показывает тот или иной отказ, а 0 - нормальное (возможно, преждевременное) завершение. Полный список значений поля Accept приведен в параграфе 3.3.

Если поле Accept отлично от 0 (в сообщении любой стороны), результаты всех сессий OWAMP-Test, собранные в этй сессии OWAMP-Control, **следует** считать непригодными, даже если Fetch-Session с SID из данной сессии работает для другой сессии OWAMP-Control. Если Accept не было передано совсем (по любой причине, включая разрыв соединения TCP, используемого для OWAMP-Control), результаты всех сессий OWAMP-Test, собранные в этй сессии OWAMP-Control, **можно** считать непригодными.

Number of Sessions указывает число описаний сессий, которые следуют сразу за заголовком Stop-Sessions.

Поле Number of Sessions **должно** указывать число сеансов передачи, начатых локальной стороной управляющего соединения, которые не были прерваны ранее командой Stop-Sessions (т. е. Control-Client **должен** учитывать каждое воспринятое сообщение Request-Session с установленным Conf-Receiver, Control-Server **должен** учитывать каждое воспринятое сообщение Request-Session с установленным Conf-Sender). Если сообщение Stop-Sessions не учитывает точно сеансы передачи, контролируемые данной стороной, он считается недействительным и соединение **следует** разорвать, а все полученные результаты считать недействительными.

Каждое описание сессии представляет одну сессию OWAMP-Test.

SID является идентификатором сессии и служит для указания описываемой сессии.

Next Seqno указывает следующий порядковый номер, который был бы отправлен из данного сеанса передачи. Для завершенных сессий это поле будет совпадать с NumPackets из Request-Session.

Number of Skip Ranges указывает число реальных пропусков в процессе передачи. Это диапазоны пакетов, которые в действительности не были отправлены передающим процессом. Например, если сеанс передачи начался слишком поздно для передачи первых 10 и это является единственным пропуском в расписании, поле Number of Skip Ranges будет иметь значение 1. Одно описание Skip Range будет иметь First Seqno Skipped = 0 и Last Seqno Skipped = 9. Дополнительная информация приведена в параграфе 4.1.

если соединение OWAMP-Control прерывается при передаче команды Stop-Sessions, получатель **может** не объявлять все результаты сессии непригодными. Он **должен** отбросить все записи для пакетов, следующих (с большим порядковым номером) за последним пакетом, который был действительно получен до всех записей о потере пакетов. Это поможет отделить пакеты, потерянные в сети, от пакетов, которые отправитель не передал.

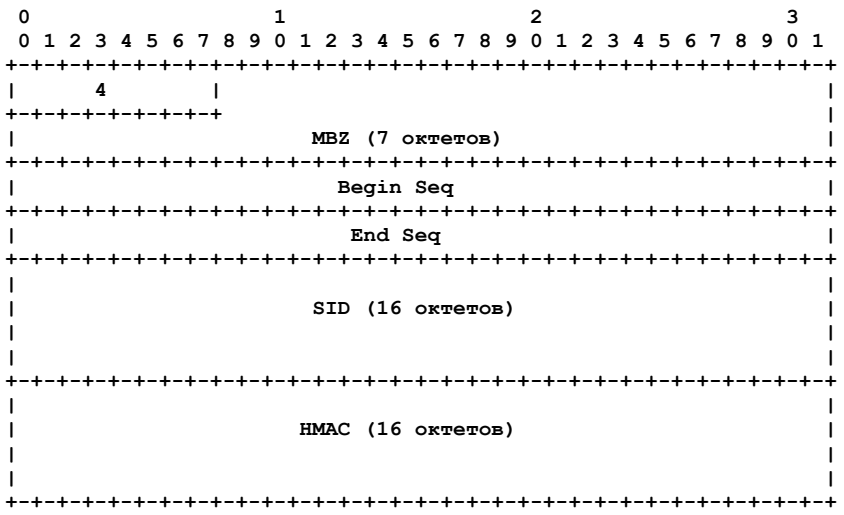
Если принимающая сторона сессии OWAMP-Test узнает из сообщения OWAMP-Control Stop-Sessions, что последний порядковый номер у отправителя OWAMP-Test менбше любого из реально полученных порядковых номеров, результаты всех сессии OWAMP-Test **должны** быть признаны недействительными.

Получатель в сессии OWAMP-Test по команде OWAMP-Control Stop-Sessions **должен** отбросить все записи для пакетов (включая записи о потерях) с (рассчитанным) временем отправки в интервале Timeout до текущего времени. Это обеспечивает статистическую согласованность измерения потерь и дубликатов в случаях, когда Timeout больше времени, требуемого на выполнение команды Stop-Sessions.

Для завершения сессии каждой стороне управляющего соединения **следует** дождаться завершения всех измерительных сессий перед отправкой сообщения Stop-Sessions. Время завершения каждой сессии определяется интервалом Timeout после запланированного времени для последнего порядкового номера. Конечные точки **могут** немного добавлять к рассчитаному времени завершения для передающей конечной точки, чтобы обеспечить прием получателем сообщения Stop-Sessions уже после интервала Timeout.

Для преждевременного завершения сессии сторона, инициирующая эту команду, **должна** остановить свой поток передачи OWAMP-Test, чтобы передать значения Session Packets Sent до отправки команды. Этой стороне **следует** дождаться получения отклика Stop-Sessions перед остановкой приема потоков, чтобы можно было использовать значения из полученного сообщения Stop-Sessions для проверки данных.

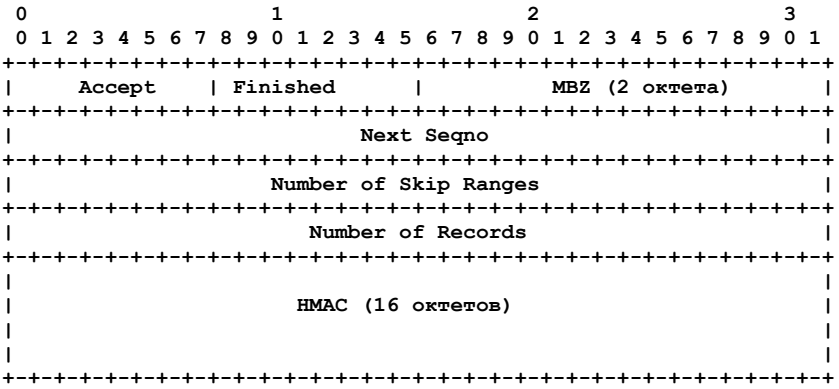
3.9. Fetch-Session



Begin Seq указывает порядковый номер первого запрошенного пакета, а End Seq - номер последнего. Если Begin Seq содержит только 0, а End Seq - только 1, это означает запрос всех пакетов сессии.

Если запрошена сессия целиком и эта сессия продолжается или прервана аномальным способом, запрос на выборку сессии **должен** отвергаться. Если запрошена неполная сессия, **следует** возвращать все пакеты, попадающие в запрошенный диапазон. Отметим, что в результате невозможности ввода команд между Start-Sessions и Stop-Sessions неполные запросы могут возникать лишь на другом соединении OWAMP-Control (от того же или другого Control-Client).

Сервер **должен** отвечать сообщением Fetch-Ack, формат которого показа ниже.



Как обычно, ненулевое значение поля Accept говорит, что команда отвергнута. В таких случаях сервер **должен** заполнить все оставшиеся поля нулями, а клиент **должен** игнорировать все эти поля (за исключением HMAC). Полный список возможных значений поля Accept приведен в параграфе 3.3.

Поле Finished имеет отличное от 0 значение, если сессия OWAMP-Test прервана.

Next Seqno указывает следующий порядковый номер, который был бы отправлен из этого сеанса передачи. Для завершенных сессий значение этого поля будет совпадать с NumPackets из Request-Session. Эта информация доступна лишь для прерванных сессий. Если Finished = 0, сервер **должен** установить Next Seqno = 0.

Number of Skip Ranges указывает число реальных пропусков в процессе передачи. Эта инфоормация доступна лишь для прерванных сессий. Если Finished = 0, сервер **должен** установить Skip Ranges = 0.

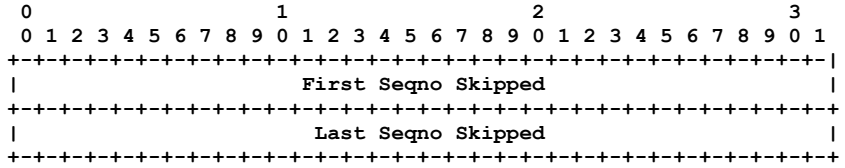
Number of Records указывает число записей для пакетов, попадающих в запрошенный диапазон. Это значение может быть меньше Number of Packets при воспроизведении команды Request-Session, поскольку сессия может быть завершена преждевременно или больше по причине наличия дубликатов.

Отличное от 0 значение Accept говорит о завершении отклика на сообщение Fetch-Session. Если Accept = 0, сервер **должен** незамедлительно передать данные соответствующей сессии OWAMP-Test.

Данные сессии OWAMP-Test являются конкатенацией перечисленных ниже значений.

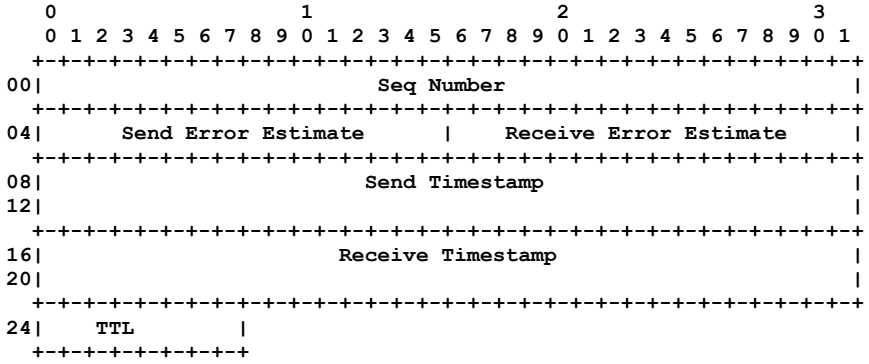
- Воспроизведение команды Request-Session, которая была использована для старта сессии, измененная так, что всегда указываются реальные номера портов отправителя и получателя в сеансе OWAMP-Test.
- Необязательные (число задано) описания Skip Range. Последний (возможно, неполный) блок (16 октетов) описания Skip Range при необходимости дополняется нулями.
- 16 октетов HMAC.
- Необязательные (число задано) записи для пакетов. Последний (возможно, неполный) блок (16 октетов) описания Skip Range при необходимости дополняется нулями.
- 16 октетов HMAC.

Описание Skip Range является просто двумя порядковыми номерами, указывающими диапазон пропущенных пакетов.



Описания Skip Range следует передавать в порядке возрастания First Seqno. Если какие-либо Skip Range перекрываются или нарушают порядок, данные сессии считаются непригодными и соединение **следует** закрывать, считая результаты недействительными.

Каждая запись для пакета имеет размер 25 октетов и включает 4-октетный порядковый номер, 8-октетную временную метку отправки, 2 октета оценки ошибки этой метки, 8-октетную временную метку приема, 2 октета оценки ошибки этой метки и 1 октет TTL или Hop Limit (IPv6).



Записи для пакетов передаются в том же порядке, как передавались реальные пакеты.

Отметим, что потерянные пакеты (при наличии потерь в сеансе OWAMP-Test) **должны** указываться в последовательности пакетов. Они могут указываться в точке обнаружения потери или в любом месте после нее. Записи для потерянных пакетов отличаются по приведенным ниже признакам.

- В поле временной метки передачи указывается предполагаемое значение (расписание передающей стороны).
- В поле оценки ошибки времени передачи указываются Multiplier=1, Scale=64, S=0 (см. описание OWAMP-Test, где приведены определения этих параметров и описан формат временных меток и оценки ошибок).
- Обычная оценка ошибки времени приема, определяемая ошибкой часов, используемых для объявления потери пакета (пакет считается потерянным, если он не получен по истечении интервала Timeout с момента предполагаемого прибытия по часам приемной стороны).
- Временная метка приема содержит только нули.
- TTL = 255.

4. OWAMP-Test

В этом разделе описывается протокол OWAMP-Test, который работает на основе UDP с использованием адресов IP и номеров портов отправителя и получателя, согласованных в процессе обмена Request-Session.

Как и OWAMP-Control протокол OWAMP-Test имеет три режима - без аутентификации, с проверкой подлинности и с шифрованием. Все сессии OWAMP-Test, созданные в сессии OWAMP-Control, наследуют режим из нее. Клиент и сервер OWAMP-Control, отправитель и получатель OWAMP-Test могут быть разными машинами (в типовом случае предполагается использование только двух машин).

4.1. Поведение отправителя

4.1.1. Синхронизация пакетов

Расписание передачи основанное на временных интервалах, описанных выше, в сочетании с запланированным временем начала сессии позволяет отправителю и получателю рассчитать точное расписание отправки пакетов независимо друг от друга. Такое планирование передачи выполняется независимо для разных сессий OWAMP-Test даже в рамках одной сессии OWAMP-Control.

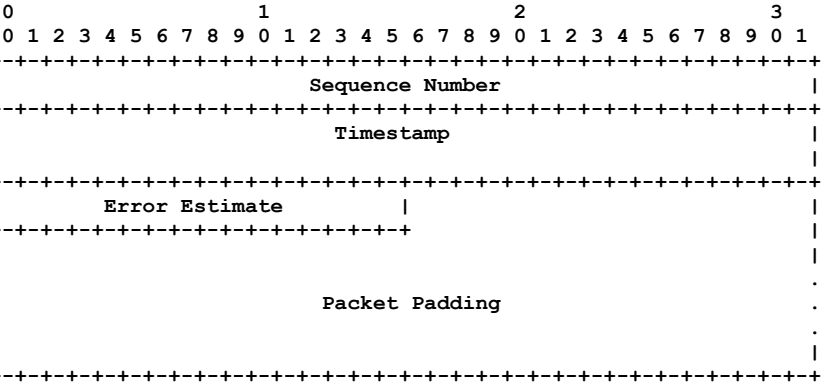
Рассмотрим любую сессию OWAMP-Test. После завершения обмена Start-Sessions отправитель готов к началу передачи пакетов. В обычных вариантах применения OWAMP время отправки первого пакета находится в ближайшем будущем (возможно, доли секунды). Отправителю **следует** передавать пакеты как можно ближе к запланированному времени с одним исключением - если запланированное время уже прошло и отделено от текущего интервалом больше Timeout, отправителю **недопустимо** передавать пакет (он в любом случае будет считаться потерянным). Отправитель **должен** отслеживать пакеты, которые не были переданы. Эта информация будет передана получателю путем установки Skip Ranges в сообщении Stop-Sessions получателю по завершении теста. Поле Skip Ranges передается также Fetch-Client как часть результатов сессии. Пропуски в запланированной передаче могут возникать, если в команде Request-Session было указано уже прошедшее время, обмен Start-Sessions занял слишком много времени или отправитель не смог своевременно начать обслуживание сессии OWAMP-Test в результате внутренних проблем планирования ОС. Пакеты из прошлого, отделенного от текущего времени интервалом меньше Timeout, **следует** передать как можно быстрее. При нормальных скоростях тестирования и значениях тайм-аута число пакетов в таких «пиках» ограничено. Тем не менее, хостам **не следует** преднамеренно планировать сессии с такими «пиками».

Независимо от каких-либо задержек планирования каждый фактически отправленный пакет **должен** иметь максимально точную временную метку отправки (в пакете).

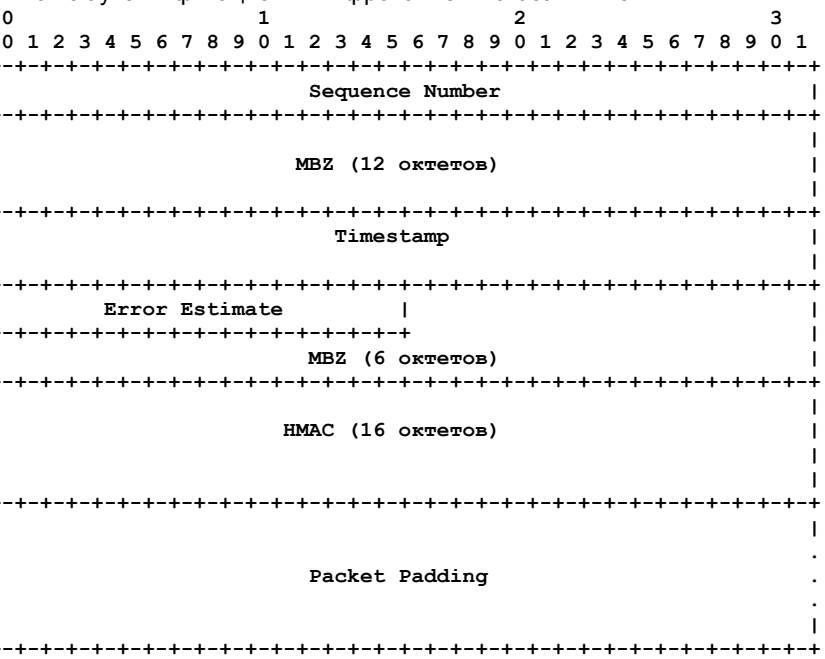
4.1.2. Формат и содержимое пакетов OWAMP-Test

Отправитель передает получателю поток пакетов по расписанию, указанному в команде Request-Session. Отправителю **следует** устанавливать в поле IPv4 TTL (или IPv6 Hop Limit) в пакетах UDP значение 255. Формат пакетов UDP в потоке зависит от используемого режима.

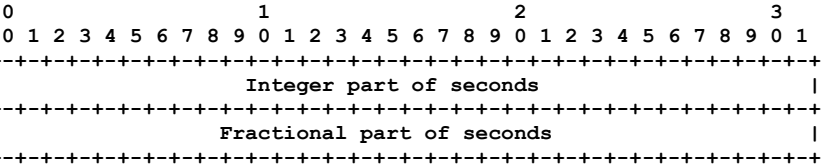
Для режима без аутентификации формат пакета имеет вид, показанный ниже.



Формат пакетов для режимов с аутентификацией и шифрованием показан ниже.

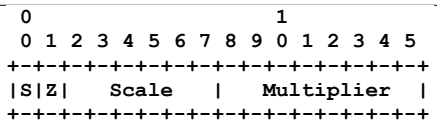


Формат временных меток совпадает с описанным в [RFC1305] и первые 32 бита представляют беззнаковым целым числом количество целых секунд, прошедших с 0 часов 1 января 1900 г., а следующие 32 бита представляют дробную часть нецелой секунды с указанного момента.



Поле Error Estimate указывает оценку ошибки синхронизации и использует показанный ниже формат.

Первый бит (S) **следует** устанавливать, если генерирующая временную метку сторона имеет часы, синхронизированные с UTC от внешнего источника (например, бит следует устанавливать при получении позиции и текущего времени от GPS или использовании NTP для синхронизации с внешним источником stratum 0). Если часы не имеют источника внешней синхронизации, устанавливать бит **не следует**. Следующий бит имеет семантику полей



MBZ, он **должен** сбрасываться отправителем и игнорироваться в остальных случаях. Следующие 6 битов указывают целое число без знака, так же как поле Multiplier. Эти поля интерпретируются как оценка ошибки выражением $Multiplier*2^{-32}*2^{Scale}$ (в секундах). В поле Multiplier **недопустимо** значение 0 и пакеты с Multiplier = 0 **следует** считать поврежденными и отбрасывать.

Порядковые номера начинаются с 0 и увеличиваются на 1 для каждого следующего пакета.

Следовательно, минимальный размер сегмента данных составляет 14 октетов в режиме без аутентификации и 48 октетов в режимах с аутентификацией и шифрованием.

Схема пакетов OWAMP-Test одинакова для режимов с аутентификацией и шифрованием. Однако операции шифрования и проверки подлинности различаются. Разница заключается в том, что в режиме шифрования порядковые номера и временные метки защищены для обеспечения максимальной защиты целостности и конфиденциальности, а в режиме с аутентификацией защищаются лишь порядковые номера а временные метки передаются в открытом виде. Открытая передача временных меток позволяет сократить время между получением временной метки отправителем и реальной передачей пакета. В режиме с шифрованием отправитель получает временную метку, шифрует ее и передает пакет, а в режиме с аутентификацией этап шифрования пропускается, что позволяет повысить точность (порядковый номер может быть зашифрован и аутентифицирован до получения временной метки).

В режиме с аутентификацией первый блок (16 октетов) каждого пакета шифруется с использованием AES (ECB) mode.

Подобно сессиям OWAMP-Control, каждая сессия OWAMP-Test имеет два ключа - AES Session-key и HMAC Session-key. Однако получение этих ключей происходит иначе - в OWAMP-Control ключи генерируются клиентом и сообщаются (как часть Token) в процессе организации соединения как часть сообщения Set-Up-Response, а в описываемом здесь OWAMP-Test ключи выводятся из ключей OWAMP-Control и SID.

OWAMP-Test AES Session-key получается путем шифрования OWAMP-Control AES Session-key (совпадает с AES Session-key в соответствующей сессии OWAMP-Control, где он применяется в другом режиме цепочки) с помощью AES с 16-октетным идентификатором SID в качестве ключа (шифрование ECB с одним блоком). Результатом является OWAMP-Test AES Session-key, применяемый для шифрования (и расшифровки) пакетов в конкретной сессии OWAMP-Test. Отметим, что все OWAMP-Test AES Session-key, OWAMP-Control AES Session-key и SID состоят из 16 октетов.

OWAMP-Test HMAC Session-key получается путем шифрования OWAMP-Control HMAC Session-key (совпадает с HMAC Session-key в соответствующей сессии OWAMP-Control) с помощью AES с 16-октетным идентификатором SID в качестве ключа (шифрование с двумя блоками CBC и IV=0). Результатом является OWAMP-Test HMAC Session-key, применяемый для проверки подлинности пакетов в отдельной сессии OWAMP-Test. Отметим, что все OWAMP-Test HMAC Session-key и OWAMP-Control HMAC Session-key состоят из 32 октетов, а SID - из 16.

Режим ECB, применяемый для шифрования первого блока пакетов OWAMP-Test в режиме с аутентификацией, не использует цепочки, поэтому потеря, дублирование или нарушение порядка пакетов не вызывают проблем при расшифровке пакетов в сессии OWAMP-Test.

В режиме с шифрованием два первых блока (32 октета) шифруются в режиме AES CBC. Ключ AES Session-key выводится так же, как для режима с аутентификацией. Каждый пакет OWAMPшифруется как отдельный поток, с одной операцией сцепки — цепочка не охватывает множество пакетов, поэтому потеря, дублирование или нарушение порядка пакетов не вызывают проблем. Вектор инициализации для шифрования CBC имеет значение 0 во всех битах.

> Примечание для разработчиков. Планирование ключей для каждой сессии OWAMP-Test **может** быть организовано лишь один раз на сессию, а не для каждого пакета.

HMAC в OWAMP-Test охватывает лишь часть пакета, которая также шифруется. Таким образом, в режиме с аутентификацией HMAC охватывает первый блок (16 октетов), в режиме шифрования — два первых блока (32 октета). В OWAMP-Test значение HMAC не шифруется (отметим, это отличие от OWAMP-Control, где применяется потоковое шифрование и блоки HMAC тоже зашифрованы).

В режиме без аутентификации шифрование или проверка подлинности не применяются.

В полях Packet Padding пакетов OWAMP-Test **следует** использовать псевдослучайные значения (они **должны** генерироваться независимо от других упоминаемых в документе псевдослучайных чисел). Однако реализация **должна** поддерживать конфигурационный параметр (опция) или другой способ для заполнения Packet Padding нулями.

Интервал между пакетами рассчитывается в соответствии с расписанием, как указано в описании команды Request-Session. Здесь не рассмотрены вопросы расчета псевдослучайных чисел с экспоненциальным распределением предсказуемым способом, поскольку оно описано в другом разделе (см. ниже).

4.2. Поведение получателя

Получателю известно, когда отпревитель будет передавать пакеты. В сообщении Request-Session задается параметр Timeout и при задержке пакета на время, превышающее Timeout пакет считается потерянным (или «похожим на потерянный»). Отметим, что в сети никогда нет гарантии потери и «потерянный» пакет может быть доставлен в любой момент. Исходная спецификация IPv4 требует доставки пакета в течение TTL секунд (максимальное значение TTL = 255) или его «недоставки». Насколько известно авторам, это требование фактически никогда не выполнялось (и оишь полная и универсальная реализация смогла бы бы гарантировать, что пакеты не перемещаются в сети более TTL секунд). В IPv6 имя этого поля было изменено на Hop Limit. Кроме того, спецификация IPv4 не указывает время прохождения пакетом последнего интервала пути.

Выбор разумного значения Timeout является проблемой, с которой сталкивается пользователь протокола OWAMP, а не разработчики. Значение порядка двух минут вполне безопасно. Отметим, что некоторые приложения (например, интерактивный «односторонний ping») могут захотеть получать данные быстрее.

При получении пакета

- фиксируется временная метка приема;
- в режиме с аутентификацией или шифрованием выполняется требуемая расшифровка и проверка подлинности (пакеты с отказом при аутентификации **должны** отбрасываться);
- сохраняются порядковый номер пакета, время отправки и получения и TTL для IPv4 (Hop Limit для IPv6) из заголовка IP для последующей передачи результатов.

Пакеты, не полученные в интервале Timeout, считаются потерянными. Они записываются с порядковым номером, предполагавшимся временем приема, нулевым временем приема и TTL (или Hop Limit) 255.

Реализациям **следует** извлекать значение TTL/Hop Limit из IP-заголовка в пакете. Если реализация не извлекает реальное значение TTL (единственной причиной этого может невозможность доступа к полю TTL в приходящих пакетах), она **должна** записывать TTL = 255.

Реально принятые пакеты записываются в порядке их получения. Записи о потерянных пакетах служат для указания времени отправки таких пакетов. Их **следует** помещать в точке фиксации потери или позднее, в частности. **можно** записывать все потерянные пакеты в самом конце.

Пакеты со временем передачи в будущем **должны** записываться как обычно без изменения временной метки их передачи, если они не отбрасываются. Временная метка из будущего говорит о расхождении часов на величину, превышающую задержку. Некоторые данные (например, вариации задержки — jitter) могут быть получены даже без знания временной разницы. Для других типов данных корректировку лучше всего проводить потребителю информации на основе полных данных сеанса измерения с возможным привлечением внешней информации.

Пакеты с порядковыми номерами, которые уже были (дубликаты) **должны** записываться как обычно (дубликаты иногда возникают в сетях IP и протокол способен измерять это).

Пакеты **должны** отбрасываться при выполнении любого из перечисленных ниже условий.

- Временная метка отличается от текущего времени в ту или иную сторону больше, чем на Timeout.
- Временная метка передачи больше чем на Timeout отличается от запланированного времени отправки в соответствии с порядковым номером пакета.
- В режиме с аутентификацией или шифрованием проверка HMAC не прошла.

5. Расчет псевдослучайных чисел

Здесь описывается способ генерации экспоненциальных случайных значений, используемых протоколом. Хотя имеется достаточно много алгоритмов генерации экспоненциальных случайных переменных, большинство из них основано на логарифмической функции в качестве примитива, что может приводить к разным значениям в зависимости от конкретной реализации математической библиотеки (math). Здесь применяется алгоритм 3.4.1.S из книги [KNUTH], в котором нет упомянутой проблемы и который гарантирует одинаковый результат в любой реализации. Алгоритм относится к семейству ziggurat, разработанному в 1970-х годах G. Marsaglia, M. Sibuya и J. H. Ahrens [ZIGG]. Логарифмическая функция в нем заменена манипуляциями с битами, которые дают на выходе экспоненциальные переменные.

5.1. Высокоуровневое описание алгоритма

Для простоты описания алгоритм сначала рассматривается с естественной интерпретацией арифметических операций. Затем приводятся точные детали типов данных, арифметики и генерации случайных переменных с однородным распределением. Это почти дословная цитата из книги [KNUTH], стр.133.

Алгоритм S. Для данного положительного действительного числа mu, создается случайная переменная с экспоненциальным распределением и средним значением mu.

Сначала заранее рассчитываются константы

```
Q[k] = (ln2)/(1!) + (ln2)^2/(2!) + ... + (ln2)^k/(k!),  1 <= k <= 11
```

Точные значения, которые **должны** применяться во всех реализациях, приведены в следующем параграфе. Это нужно для того, чтобы все реализации давали одинаковые псевдослучайные последовательности.

S1. [Получение U и сдвиг] Генерируется 32-битовая случайная дробная часть с однородным распределением

```
U = (.b0 b1 b2 ... b31)     [обратите внимание на точку в начале]
```

Находится первый нулевой бит b_j и выполняется сдвиг начальных j+1 битов путем установки U <- (.b_{j+1} ... b31)

Примечание. В редких случаях отсутствия нулевого бита алгоритм возвращает mu*32*ln2.

S2. [Подходит сразу?] Если U < ln2, устанавливается X <- mu*(j*ln2 + U) и алгоритм завершает работу (Q[1] = ln2.)

S3. [Минимизация.] Находится наименьшее k >= 2 такое, что U < Q[k]. Генерируется k новых дробных частей с однородным распределением U1,...,Uk и устанавливается V <- min(U1,...,Uk).

S4. [Выдача результата] Устанавливается X <- mu*(j + V)*ln2.

5.2. Типы данных, представление и арифметика

Алгоритм верхнего уровня работает с действительными числами, обычно представленными в форме с плавающей запятой (точкой). Данная спецификация предписывает использовать взамен 64-битовые целые числа без знака.

Целые числа u_int64_t интерпретируются как действительные путем установки запятой (точки) после первых 32 битов. Иными словами, интерпретация имеет форму отображения

```
u_int64_t u;
```

```
u  |--> (double)u / (2**32)
```

Алгоритм создает последовательность таких целых чисел u_int64_t для любого заданного значения SID, гарантируя одинаковые последоватльности в любой реализации.

Представления первых 11 значений u_int64_t в массиве Q алгоритма верхнего уровня **должны** иметь значения:

```
#1      0xB17217F8,
#2      0xEEF193F7,
#3      0xFD271862,
#4      0xFF9D6DD0,
#5      0xFFF4CFD0,
#6      0xFFFEE819,
#7      0xFFFFE7FF,
#8      0xFFFFFE2B,
#9      0xFFFFFFE0,
#10     0xFFFFFFFE,
#11     0xFFFFFFFF
```

Например, Q[1] = ln2 действительно интерпретируется как 0xB17217F8/(2**32) = 0,693147180601954, а для j > 11, Q[j] = 0xFFFFFFFF.

Небольшое целое число j в алгоритме верхнего уровня представляется как значение $j*2^{32}$ типа u_int64_t.

Операции сложения выполняются как обычно с целыми числами u_int64_t, однако умножение в алгоритме верхнего уровня следует заменить операцией

```
(u, v) |---> (u * v) >> 32.
```

Реализации **должны** точно рассчитывать произведение u*vy. Например, для этого может применяться часть арифметики 128-битовых целых чисел без знака (см. пример реализации в приложении A).

5.3. Однородные случайные величины

Процедура получения последовательности 32-битовых случайных чисел (например, U в алгоритме S) основана на использовании алгоритма AES в режиме счетчика. Для точного описания работы алгоритма применяется два примитива Rijndael. Из прототипы и спецификация даны ниже и предполагается, что они будут представлены поддерживающей Rijndael реализацией, такой как [RIJN].

- Функция, инициализирующая ключ Rijndael байтами из seed (идентификатор SID), имеет вид

```
void KeyInit(unsigned char seed[16]);
```

- Функция, шифрующая 16-октетный блок inblock с помощью заданного ключа, возвращая 16-октетный шифрованный блок (keyInstance является «непрозрачным» типом, представляющим ключи Rijndael) имеет вид

```
void BlockEncrypt(keyInstance key, unsigned char inblock[16]);
```

Алгоритм Unif. Для данного 16-октетного значения seed создается последовательность 32-битовый случайных чисел без знака с однородным распределением. В OWAMP в качестве seed применяется значение SID (идентификатор сессии) из протокола управления.

U1. [Инициализация ключа Rijndael] key <- KeyInit(seed) [Инициализация 16-октетного беззнакового счетчика с сетевым порядком байтов] c <- 0

U2. [Требуются дополнительные случайные байты?] Set i <- c mod 4. If (i == 0) set s <- BlockEncrypt(key, c)

U3. [Инкрементирование счетчика как 16-октетного числа без знака] c <- c + 1

U4. [Вывод] Выводится i_th квартетов октетов из s, начиная со старших октетов, с преобразованием к естественному порядку байтов и представлением в виде значения OWPNum64 (как в 3.b).

U5. [Цикс=л] переход к U2.

6. Вопросы безопасности

6.1. Введение

Цель режима с аутентификацией заключается в обеспечении с помощью одной парольной фразы услуг, предоставляемых сервером OWAMP-Control. Это может быть полезно при разных обстоятельствах. Режим с аутентификацией предназначен для предотвращения несанкционированного доступа к услугам.

Дополнительной целью введения этого режима было предотвращение возможности подделки результатов теста без воздействия на остальной трафик злоумышленником на пути между отправителем и получателем OWAMP-Test, не способным читать тестовый трафик.

Цель режима с шифрованием несколько иная — он осложняет посторонним на пути через сеть возможность представить результаты Лучше», чем на самом деле. Это особенно важно в тех случаях, когда клиент или сервер не является ни отправителем, ни получателем.

Шифрование OWAMP-Control в режиме AES CBC с блоками HMAC после каждого сообщения нацелено на (i) обеспечение конфиденциальности обмена и (ii) проверки подлинности каждого сообщения.

6.2. Предотвращение атак на службы

Сессии OWAMP-Test, нацеленные на ничего не подозревающую систему, могут служить для организации DoS[1]-атак. В режиме без аутентификации серверам **следует** ограничивать получателей хостами, которые они контролируют или клиентами OWAMP-Control client.

[1]Denial of service — отказ в обслуживании.

Если в настройках не задано иное, серверы по умолчанию **должны** отвергать запросы, в которых поле Receiver Address отличается от адреса, через который организовано управляющее соединение, или адреса сервера (хоста, которым сервер управляет). С учетом процедуры согласования TCP и порядковых номеров управляющего соединения это гарантирует, что передающие такие запросы хосты реально являются теми, за кого себя выдают или хотя бы находятся на пути к ним. Если любая из проверок или процедура согласования была пропущена, злоумышленник в любой точке Internet может запросить отправку большого числа пакетов на хост его жертвы.

В любом случае пакеты OWAMP-Test с данным адресом отправителя **должны** передаваться лишь с узла, которому этот адрес назначен (т. е. подмена адресов не разрешается).

6.3. Скрытые информационные каналы

Сессии OWAMP-Test могут использоваться в качестве скрытых каналов передачи информации. Это обстоятельство следует учитывать в средах, где возможность утечки имеет значение.

6.4. Требование включения AES в реализации

Отметим, что для генерации псевдослучайных чисел применяется алгоритм AES в режиме счетчика (counter mode), поэтому реализация AES **должна** быть включена даже для серверов, поддерживающих лишь работу без аутентификации.

6.5. Ограничение использования ресурсов

Сервер OWAMP может потреблять ресурсы разных типов, наиболее важными из которых являются пропускная способность сети и память (первичная и вторичная) для хранения результатов тестирования.

Все реализации серверов OWAMP **должны** включать технические механизмы для ограничения используемой пропускной способности и памяти. Механизмы управления ресурсами, потребляемыми пользователями без аутентификации и аутентифицированными с помощью KeyID и парольных фраз, **следует** разделять. Принятая по умолчанию конфигурации **должна** включать эти механизмы и устанавливать разумно низкие пределы использования ресурсов.

Одним из способов реализации механизмов ограничения ресурсов является привязка каждой сессии к классу пользователей. Классы частично упорядочиваются отношениями «включения», один класс (все пользователи) всегда присутствует и включает все остальные классы. Привязка сессии к классу пользователей может основываться на наличии аутентификации сессии, KeyID, диапазоне адресов IP, времени суток или иных факторах. Для каждого класса задается предел используемых ресурсов пропускной способности (в бит/с) и памяти для хранения результатов (в октетах). octets). Наряду с ограничением сервер будет отслеживать текущее использование ресурсов. Когда сеанс запрашивается пользователем определенного класса, рассчитываются требуемые для сессии ресурсы - средняя пропускная способность (на основе планирования передачи) и максимальное использование памяти (на основе числа пакетов и октетов каждого пакета, которые нужно хранить[1]). Эти параметры использования ресурсов добавляются к текущим параметрам загрузки ресурсов для данного класса пользователей. Если полученные значения выходят за установленные для данного класса пределы, сессия отвергается. При возврате ресурсов добавленные ранее значения вычитаются из параметров загрузки ресурсов для данного класса. Пропускная способность возвращается по завершении сеанса, а память - после удаления данных. Для сессий без аутентификации потребляемую сессией OWAMP-Test память **следует** возвращать после закрытия (любым способом) соединения OWAMP-Control, инициировавшего данную сессию. Для сессий с аутентификацией администратору, настраивающему сервис, следует давать возможность установить точные правила. Полезными для реализации **могут** быть механизмы, способные автоматически возвращать память по истечении настраиваемого (на основе класса пользователей) интервала после завершения сессии OWAMP-Test.

6.6. Использование криптографических примитивов в OWAMP

На начальном этапе разработки протокола рассматривалось использование TLS[2] [RFC2246, RFC3546] и IPsec [RFC2401] в качестве механизмов криптографической защиты OWAMP, а позднее был рассмотрен протокол DTLS. Недостатки этих механизмов частично указаны ниже.

TLS

- TLS можно применить для защиты TCP-соединений OWAMP-Control, но сложно использовать для UDP-сессий OWAMP-Test. При потере пакетов OWAMP-Test они не повторяются, поэтому пакету нужно аутентифицировать и (опционально) шифровать по-отдельности для сохранения применимости каждого пакета. Потоковый сервис TLS сложно приспособить для этого.
- Работая с потоком, TLS не аутентифицирует отдельные сообщения (даже в OWAMP-Control). Самым простым способом было бы добавление того или иного общеизвестного заполнения в каждое сообщение и проверки неизменности этого заполнения перед использованием сообщения. Это приводит к частично утрате привлекательности решения (просто использовать TLS). Кроме того будет сложнее оценить безопасность этой схемы при разных режимах и опциях TLS - безопасность почти наверняка будет обеспечиваться не во всех случаях. Способность атакующего заменять части сообщения (а именно, конец) случайным мусором может оказать существенное влияние на безопасность и потребует тщательного анализа. Предположим, например, что используемый для управления скоростью параметр будет заменен случайным мусором - вероятность того, что это число (как целое без знака) будет достаточно большим совсем не мала.
- В зависимости от режима использования могут потребоваться сертификаты для всех пользователей и PKI. Даже принимая желательность PKI, применять это сегодня просто нереально.
- TLS требует достаточно большой реализации. Например, OpenSSL превышает реализацию всего OWAMP. Это может вызвать проблемы для встраиваемых систем.

[1]Отметим, что исходящие сессии не требуют использования памяти.
[2]Transport Layer Security - защита на транспортном уровне.

DTLS

- Дублирование и нарушение порядка - это те явления, которые OWAMP должен измерять, поэтому меры защиты от повторов (anti-replay) и нарушения порядка в DTLS будут препятствовать измерениям, поскольку пакеты не попадут в соответствующие части кода OWAMP. Конечно, можно изменить DTLS так, чтобы эти меры были ослаблены или даже задать проверку сообщений в тщательно продуманной последовательности где-то между проверками DTLS, но тогда преимущества этого протокола просто не будут использоваться.

- В режиме с аутентификацией временные метки передаются в открытом виде и не защищены криптографически, тогда как остальная часть сообщения имеет такую же защиту как в режиме с шифрованием. Это является компромиссом с отказом от криптографической в пользу точности временных меток. Например, аппаратная реализация OWAMP APAN [APAN] может поддерживать режим с аутентификацией. Точность измерений составляет доли микросекунд. Ошибки в измерениях OWAMP реализации Abilene [Abilene] (программная реализация с шифрованием) превосходят 10 мксек. Пользователи в разных средах сталкиваются с различными проблемами и для некоторых микросекундная точность может быть важна. В то же время пользователи этих же систем могут быть озабочены контролем доступа к сервису. Режим с аутентификацией позволяет им контролировать доступ к серверу с одновременным использованием незащищенных временных меток, возможно создаваемых на аппаратном уровне.

IPsec

- То, что названо здесь режимом с аутентификацией, невозможно (IPsec не позволяет аутентифицировать часть пакета).

- Развертывание путем IPsec и OWAMP может быть раздельным, если OWAMP не зависит от IPsec. После 9 лет применения IPsec, лиль 0,05% трафика в крупных магистральных сетях, таких как Abilene, использует IPsec (для сравнения SSH применяется в 2-4%, а HTTPS - 0,2-0,6%). Желательно иметь возможность развертывания OWAMP на максимально большом числе разных платформ.

- Проблемы развертывания протокола, зависящего от IPsec, обостряются для облегченных встраиваемых устройств. Коммутаторы Ethernet, «модемы» DSL и другие устройства часто не поддерживают IPsec.

- API для работы с IPsec из приложений в настоящее время малопонятны. Создание программы, которая должна шифровать некоторые пакеты, аутентифицировать некоторые пакеты и не трогать остальные (для того же получателя), значительно сложнее в IPsec, нежели просто измерения в IP.

По указанным причинам было принято решение применять криптографический протокол (на базе блочного шифра в режиме CBC), отличающийся от TLS и IPsec.

6.7. Замена криптографического примитива

В будущем может возникнуть потребность в замене алгоритма AES или способа его использования в OWAMP другим криптографическим примитивом или внести в протокол другие изменения, связанные с защитой. OWAMP обеспечивает четко указанные точки расширения - слово Modes в приветствии сервера и отклик Mode в сообщении Set-Up-Response. Например, при необходимости заменить AES другим блочным шифром с размером блока 128 битов это можно сделать достаточно просто. Берутся два бита из резервной (MBZ) части слова Modes в приветствии сервера, один из этих битов применяется для индикации режима шифрования с новым алгоритмом, а другой - для режима с аутентификацией, использующей новый шифр. На практике можно обойтись даже одним битом, если клиенту разрешено возвращать выбор режима с несколькими установленными битами, один из которых будет указывать поддержку нового шифра (в случае сервера) или выбор (в случае клиента) и продолжение использования уже выделенных режимов аутентификации и шифрования. Такая оптимизация концептуально не важна, но может обеспечить на практике более эффективное использование битов. Затем, если новый шифр согласован, все последующие операции просто используют его вместо AES. Отметим, что в этом случае будет применяться обычная последовательность перехода - реализации вероятно начнут с поддержки и предпочтения нового шифра, а затем прекратят поддержку старого (например, сочтенного небезопасным).

При необходимости более обширных изменений (возмоно, для замены AES на шифр с 256-битовыми блоками) ситуация усложняется и потребуется менять схему сообщений. Однако изменения все еще могут быть выполнены в рамках расширения схемы OWAMP с использованием слов Modes и Mode. Семантика новых битов (или одного бита при использовании описанной выше оптимизации) будет включать изменение схемы сообщения, а также смену криптографического примитива.

Каждый из битов слова Modes можно использовать как независимое расширение. Расширения, указанные разными битами, ортогональны. Например, один бит может быть выделен для замены AES-128 другим шифром, другой - для добавления свойства протокола (например, поддержки групповой адресации), еще один - для смены функции создания ключей и т. д. Изменение версий не является линейным процессом, но частично упорядочено. Реализация может поддерживать любое подмножество этих свойств (некоторые из них могут быть обязательными, например, переход на более защищенный шифр).

Если потребуется шифр с иным размером ключа (скажем, 256 битов), нужна будет и новая функция вывода ключей для OWAMP-Test. Семантику смены шифра **следует** в таком случае связывать с семантикой замены функции вывода ключей KDF[1]. Одной из KDF для таких целей может служить псевдослучайная функция (PRF[2]) с выходом подходящего размера, таким как 256 битов (возможно, HMAC-SHA256, если она еще будет считаться защищенной PRF), который можно применить для создания сеансовых ключей OWAMP-Test на основе сеансового ключа OWAMP-Control с использованием его в качестве ключа HMAC и идентификатора SID в качестве сообщения HMAC.

Отметим, что описанная выше схема замены уязвима для атак на понижение возможностей и атакующий на пути передачи может поменять биты режима при согласовании, чтобы применялся самый старый и слабый режим из числа поддерживаемых обеими сторонами. Если это существенно в процессе замены шифра, схема может быть дополнена

[1]Key derivation function.
[2]Pseudo-random function.

мерами защиты от таких атак (возможно путем повторного обмена режимами после организации защищенного соединения, сравнения двух наборов слов режима и сброса соединения при несоответствии).

6.8. Долгосрочные ключи с ручным управлением

OWAMP-Control использует долгосрочные ключи с ручным управлением. Эти ключи применяются для автоматического согласования сеансовых ключей в каждой сессии OWAMP-Control, работающей в режиме с аутентификацией или шифрованием. Число таких ключей, управляемых сервером, линейно растет (и фактически совпадает) с числом разных пользователей (людей, ролей или роботов, представляющих сайты), которым нужно соединение с этим сервером. Аналогично, число поддерживаемых вручную ключей у клиента совпадает с числом серверов, к которым клиенту нужно подключиться. Применение долгосрочных ключей с ручным управлением соответствует [BCP107].

6.9. Отказ от использования времени в качестве «затравки»

Вполне естественно использовать текущее время в качестве «затравки» (salt) при создании сеансовых ключей. К сожалению это оказывается слишком ограниченным.

Хотя OWAMP часто работает на хостах с хорошо синхронизированными часами, возможна работа протокола и на хостах с совершенно не синхронизированными часами. Полученные таким образом задержки, естественно, не могут применяться напрямую. Однако некоторые параметры, такие как односторонние потери, изменение порядка, показатели перегрузки (например, средняя задержка за вычетом минимальной) и многие другие могут применяться напрямую и незамедлительно (и улучшать данные, которые были бы получены при двухстороннем измерении). Кроме того, даже информация о задержке может быть полезна при соответствующей пост-обработке. Действительно, можно даже утверждать, что работа без надежных часов и пост-обработка результатов измерений будет обеспечивать высокую точность, поскольку доступно больше информации и возможно сопоставление данных от различных хостов при пост-обработке, но без синхронизации часов.

С учетом этого время не применяется в качестве затравки при создании ключей.

6.10. Использование AES-CBC и HMAC

Конфиденциальность OWAMP опирается на AES-CBC, а проверка подлинности сообщений - на HMAC-SHA1, урезанный до 128 битов. Случайный выбор IV важен для предотвращения атак codebook на первый блок (следует также отметить, что, благодаря 128-битовым блокам, алгоритм AES более устойчив к codebook-атакам, чем шифры с более короткими блоками). В любом случае здесь применяется случайное значение IV.

Значения HMAC **должны** проверяться. Важно выполнить эту проверку до использования сообщения, поскольку иначе будут возможны подделки. Полное сообщение, для которого проверка HMAC привела к отказу, **должны** отбрасываться (как короткие из нескольких блоков, так и длинные сообщения, такие как отклики на команду Fetch-Session). Если такое сообщение является частью сессии OWAMP-Control, соединение **должно** разрываться.

Поскольку сообщения OWAMP имеют разное число блоков, возникает проблема атак с подменой, описанных в примере 9.62 работы [MENEZES]. Для предотвращения таких атак (и упрощения реализации) размер любого сообщения раскрывается лишь после расшифровки первого блока сообщения.

Особым случаем является первое (фиксированного размера) сообщение, переданное клиентом. Здесь маркер представляет собой конкатенацию 128-битового вызова (challenge), передаваемого сервером в открытом виде, 128-битового ключа AES Session-key (генерируется клиентом, шифруется AES-CBC с IV=0) и 256-битового ключа HMAC-SHA1 Session-key, служащего для аутентификации. Поскольку IV=0, challenge (один блок шифра) просто шифруется с секретным ключом. Поэтому протокол полагается на устойчивость AES к атакам по выбранному открытому тексту (когда атакующий может подменить challenge). Следует отметить, что число блоков выбранного атакующим открытого текста, которые он может зашифровать с помощью секретного ключа, ограничено количеством сессий, которые клиент хочет инициировать. Атакующий, который знает шифрование серверного вызова (challenge) может подделать сеансовый ключ и в результате нарушить сессию, однако любой атакующий может прервать сессию путем повреждения протокольных сообщений. Поэтому никакой новой угрозы здесь не возникает, но тем не менее требуется, чтобы сервер не применял дважды один и тот же вызов (challenge). При случайной генерации challenge повтор может происходить в среднем после 2^{64} сессий, что представляется удовлетворительным, поскольку даже для очень сильно загруженных серверов, поддерживающих миллионы сессий в секунду, повтор случиться через 500 веков. Что касается второй части маркера, атакующий может создать поддельный сеансовый ключ, изменив вторую часть маркера клиента, не затрагивая первую. Такая подделка будет сразу же обнаружена клиентом, когда проверка HMAC в следующем сообщении от сервера (восприятие или отклонение соединения) завершится отказом.

7. Благодарности

Авторы благодарны Guy Almes, Mark Allman, Jari Arkko, Hamid Asgari, Steven Van den Berghe, Eric Boyd, Robert Cole, Joan Cucchiara, Stephen Donnelly, Susan Evett, Sam Hartman, Kaynam Hedayat, Petri Helenius, Scott Hollenbeck, Russ Housley, Kitamura Yasuichi, Daniel H. T. R. Lawson, Will E. Leland, Bruce A. Mah, Allison Mankin, Al Morton, Attila Pasztor, Randy Presuhn, Matthew Roughan, Andy Scherrer, Henk Uijterwaal и Sam Weiler за их комментарии, предложения, обзоры, дискуссии и правки.

8. Взаимодействие с IANA

Агентство IANA выделило номер порта TCP 861 для компоненты OWAMP-Control протокола OWAMP.

9. Поддержка естественных языков

Протокол не переносит какой-либо информации на естественных языках за исключением KeyID в OWAMP-Control с кодировкой UTF-8.

10. Литература

10.1. Нормативные документы

[AES] Advanced Encryption Standard (AES), http://csrc.nist.gov/encryption/aes/

[BCP107] Bellovin, S. and R. Housley, "Guidelines for Cryptographic Key Management", BCP 107, RFC 4107, June 2005.

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

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

[RFC2330] Paxson, V., Almes, G., Mahdavi, J., and M. Mathis, "Framework for IP Performance Metrics", RFC 2330, May 1998.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

[RFC2679] Almes, G., Kalidindi, S., and M. Zekauskas, "A One-way Delay Metric for IPPM", RFC 2679, September 1999.

[RFC2680] Almes, G., Kalidindi, S., and M. Zekauskas, "A One-way Packet Loss Metric for IPPM", RFC 2680, September 1999.

[RFC2836] Brim, S., Carpenter, B., and F. Le Faucheur, "Per Hop Behavior Identification Codes", RFC 2836, May 2000.

[RFC2898] Kaliski, B., "PKCS #5: Password-Based Cryptography Specification Version 2.0", RFC 2898, September 2000.

10.2. Дополнительная литература

[APAN] Z. Shu and K. Kobayashi, "HOTS: An OWAMP-Compliant Hardware Packet Timestamper", In Proceedings of PAM 2005, http://www.springerlink.com/index/W4GBD39YWC11GQTN.pdf

[BRIX] Brix Networks, http://www.brixnet.com/

[ZIGG] J. H. Ahrens, U. Dieter, "Computer methods for sampling from the exponential and normal distributions", Communications of ACM, volume 15, issue 10, 873-882, 1972. http://doi.acm.org/10.1145/355604.361593

[MENEZES] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, Handbook of Applied Cryptography, CRC Press, revised reprint with updates, 1997.

[KNUTH] D. Knuth, The Art of Computer Programming, vol.2, 3rd edition, 1998.

[Abilene] One-way Latency Measurement (OWAMP), http://e2epi.internet2.edu/owamp/

[RIJN] Reference ANSI C Implementation of Rijndael, http://www.esat.kuleuven.ac.be/~rijmen/rijndael/rijndaelref.zip

[RIPE] RIPE NCC Test-Traffic Measurements home, http://www.ripe.net/test-traffic/.

[SURVEYOR] Surveyor Home Page, http://www.advanced.org/surveyor/.

[SURVEYOR-INET] S. Kalidindi and M. Zekauskas, "Surveyor: An Infrastructure for Network Performance Measurements", Proceedings of INET'99, June 1999. http://www.isoc.org/inet99/proceedings/4h/4h_2.htm

[RFC1305] Mills, D., "Network Time Protocol (Version 3) Specification, Implementation and Analysis", RFC 1305, March 1992.

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[RFC2401] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[RFC3546] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J., and T. Wright, "Transport Layer Security (TLS) Extensions", RFC 3546, June 2003.

[RFC4086] Eastlake, D., 3rd, Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

Приложение A. Пример кода C для экспоненциальных переменных

Массив Q[] содержит точные значения, которые **должны** использоваться всеми реализациями (см. параграфы 5.1 и 5.2). Данное приложение дано лишь для иллюстрации.

```
/*
** Пример использования - генерируется поток экспоненциальных (mean 1)
** случайных величин (с игнорированием проверки ошибок при инициализации).
** Если нужна переменная со средним значением mu, отличным от 1, выход
** алгоритма можно умножить на mu в соответствии с правилами описываемой
** арифметики.

** Предположим, что 16-октетная «затравка» (seed) инициализирована
** (например, в качестве общего секрета OWAMP) значением
** unsigned char seed[16];

** OWPrand_context next;

** (инициализация состояния)
** OWPrand_context_init(&next, seed);

** (генерация последовательности экспоненциальных переменных)
** while (1) {
```

```
**     u_int64_t num = OWPexp_rand64(&next);
       <операции с num>
                    ...
** }
*/

#include <stdlib.h>

typedef u_int64_t u_int64_t;

/* (K - 1) является первым k, для которого Q[k] > 1 - 1/(2^32). */
#define K 12

#define BIT31   0x80000000UL    /* проверка того, что первый из младших
                                   32 битов имеет значение 0. */
#define MASK32(n)       ((n) & 0xFFFFFFFFUL)

#define EXP2POW32       0x100000000ULL

typedef struct OWPrand_context {
        unsigned char counter[16];/* Счетчик (сетевой порядок байтов).*/
        keyInstance key;          /* Ключ для шифрования счетчика.*/
        unsigned char out[16];    /* Зашифрованный блок.*/
} OWPrand_context;

/*
** Массив рассчитывается по формуле
**
**       Q[k] = (ln2)/(1!) + (ln2)^2/(2!) + ... + (ln2)^k/(k!)
**
** как описано в алгоритме S. (Тприведенные ниже значения были
** умножены на 2^32 и округлены до ближайшего целого числа.)
** Эти точные значения ДОЛЖНЫ применяться так, чтобы разные
** реализации давали одинаковые последовательности.
*/
static u_int64_t Q[K] = {
        0,        /* Заменитель, чтобы индексы массива начинались с 1. */
        0xB17217F8,
        0xEEF193F7,
        0xFD271862,
        0xFF9D6DD0,
        0xFFF4CFD0,
        0xFFFEE819,
        0xFFFFE7FF,
        0xFFFFFE2B,
        0xFFFFFFE0,
        0xFFFFFFFE,
        0xFFFFFFFF
};

/* Этот элемент представляет ln2 */
#define LN2 Q[1]

/*
** Преобразует 32-битовое целое число без знака в u_int64_t.
*/
u_int64_t
OWPulong2num64(u_int32_t a)
{
        return ((u_int64_t)1 << 32) * a;
}

/*
** Арифметические функции для чисел u_int64_t.
*/

/*
** Сложение.
*/
u_int64_t
OWPnum64_add(u_int64_t x, u_int64_t y)
{
        return x + y;
}

/*
** Умножение. Допускается переполнение. Простая реализация
** алгоритма 4.3.1.M (стр. 268) из [KNUTH].
*/
u_int64_t
OWPnum64_mul(u_int64_t x, u_int64_t y)
{
        unsigned long w[4];
        u_int64_t xdec[2];
        u_int64_t ydec[2];
```

```
        int i, j;
        u_int64_t k, t, ret;

        xdec[0] = MASK32(x);
        xdec[1] = MASK32(x>>32);
        ydec[0] = MASK32(y);
        ydec[1] = MASK32(y>>32);

        for (j = 0; j < 4; j++)
                w[j] = 0;

        for (j = 0; j < 2; j++) {
                k = 0;
                for (i = 0; ; ) {
                        t = k + (xdec[i]*ydec[j]) + w[i + j];
                        w[i + j] = t%EXP2POW32;
                        k = t/EXP2POW32;
                        if (++i < 2)
                                continue;
                        else {
                                w[j + 2] = k;
                                break;
                        }
                }
        }
        ret = w[2];
        ret <<= 32;
        return w[1] + ret;
}


/*
** «Затравка» генератора случайных чисел с использованием 16-байтового
** значения seed (идентификатор сессии вOWAMP). Эта функция реализует
** этап U1 алгоритма Unif.
*/

void
OWPrand_context_init(OWPrand_context *next, unsigned char *seed)
{
        int i;

        /* Инициализация ключа */
        rijndaelKeyInit(next->key, seed);

        /* Инициализация счетчика нулями */
        memset(next->out, 0, 16);
        for (i = 0; i < 16; i++)
                next->counter[i] = 0UL;
}


/* Функции генерации случайных чисел. */
/*
** Создает и возвращает 32-битовые случайные значения с однородным
** распределением (сохраняются в младшей половине the u_int64_t).
** Эта функция реализует этапы U2-U4 алгоритма Unif.
*/
u_int64_t
OWPunif_rand64(OWPrand_context *next)
{
        int j;
        u_int8_t  *buf;
        u_int64_t  ret = 0;

        /* Этап U2 */
        u_int8_t i = next->counter[15] & (u_int8_t)3;
        if (!i)
                rijndaelEncrypt(next->key, next->counter, next->out);

        /* Этап U3. Инкрементирование next.counter как 16-октетного
           значения в сетевым порядком байтов для AES в режиме счетчика. */
        for (j = 15; j >= 0; j--)
                if (++next->counter[j])
                        break;

        /* Этап U4. Вывод. Последние 4 байта ret содержат случайное
           целое число с сетевым порядком байтов. */
        buf = &next->out[4*i];
        for (j=0; j<4; j++) {
                ret <<= 8;
                ret += *buf++;
        }
        return ret;
}
```

```
/*
** Генерация экспоненциальной переменной со средним значением 1.
*/
u_int64_t
OWPexp_rand64(OWPrand_context *next)
{
        unsigned long i, k;
        u_int32_t j = 0;
        u_int64_t U, V, J, tmp;

        /* Этап S1. Получение U и сдвиг */
        U = OWPunif_rand64(next);

        while ((U & BIT31) && (j < 32)) { /* Сдвиг до первого 0. */
                U <<= 1;
                j++;
        }
        /* Удаление 0. */
        U <<= 1;

        U = MASK32(U);  /* Сохранение дробной части. */
        J = OWPulong2num64(j);

        /* Этап S2. Принимается сразу? */
        if (U < LN2)        /* возврат (j*ln2 + U) */
                return OWPnum64_add(OWPnum64_mul(J, LN2), U);

        /* Этап S3.  Минимизация. */
        for (k = 2; k < K; k++)
                if (U < Q[k])
                        break;
        V = OWPunif_rand64(next);
        for (i = 2; i <= k; i++) {
                tmp = OWPunif_rand64(next);
                if (tmp < V)
                        V = tmp;
        }

        /* Этап S4. Возврат (j+V)*ln2 */
        return OWPnum64_mul(OWPnum64_add(J, V), LN2);
}
```

Приложение B. Тестовые векторы

Важно, чтобы расписания тестов, создаваемые разными реализациями при дентичных входных данных, были идентичны. Нетривиальной задачей является генерация псевдослучайных переменных с экспоненциальным распределением. Чтобы помочь разработчикам в проверке совместимости, здесь представлено несколько тестовых векторов. Для каждого из приведенных 4 приведенных 128-битовых значения SID в шестнадцатеричном формате, генерируется 1 миллион 64-битовых переменных с экспоненциальным распределением, как описано выше. По мере генерации значения складываются и сумма всех 1 000 000 значений представляется в виде шестнадцатеричного числа для каждого SID. Реализации **должны** выдавать именно такие шестнадцатеричные числа. Чтобы помочь при проверке преобразования этих чисел в секунды задержки, приведены приблизительные значения (для lambda = 1). Реализациям **следует** выдавать число секунд, близкое к приведенным ниже значениям.

```
SID = 0x2872979303ab47eeac028dab3829dab2
SUM[1000000] = 0x000f4479bd317381 (1000569,739036 сек.)

SID = 0x0102030405060708090a0b0c0d0e0f00
SUM[1000000] = 0x000f433686466a62 (1000246,524512 сек.)

SID = 0xdeadbeefdeadbeefdeadbeefdeadbeef
SUM[1000000] = 0x000f416c8884d2d3 (999788,533277 сек.)

SID = 0xfeed0feed1feed2feed3feed4feed5ab
SUM[1000000] = 0x000f3f0b4b416ec8 (999179,293967 сек.)
```

Адреса авторов

Stanislav Shalunov

Internet2

1000 Oakbrook Drive, Suite 300

Ann Arbor, MI 48104

EMail: shalunov@internet2.edu

WWW: http://www.internet2.edu/~shalunov/

Benjamin Teitelbaum

Internet2

1000 Oakbrook Drive, Suite 300

Ann Arbor, MI 48104

EMail: ben@internet2.edu

WWW: http://people.internet2.edu/~ben/

Anatoly Karp

Computer Sciences Department

University of Wisconsin-Madison

Madison, WI 53706

EMail: akarp@cs.wisc.edu

Jeff W. Boote

Internet2

1000 Oakbrook Drive, Suite 300

Ann Arbor, MI 48104

EMail: boote@internet2.edu

Matthew J. Zekauskas

Internet2

1000 Oakbrook Drive, Suite 300

Ann Arbor, MI 48104

EMail: matt@internet2.edu

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru

Подтверждение

Финансирование функций RFC Editor обеспечено IETF Administrative Support Activity (IASA).