Internet Engineering Task Force (IETF)
Request for Comments: 8991
Category: Informational
ISSN: 2070-1721

B. Carpenter
Univ. of Auckland
B. Liu, Ed.
Huawei Technologies
W. Wang
X. Gong
BUPT University

May 2021

GeneRic Autonomic Signaling Protocol Application Program Interface (GRASP API)

Интерфейс GRASP API

Аннотация

Этот документ содержит концептуальную схему интерфейса с прикладными программами (Application Programming Interface или API) для базового протокола автоматической сигнализации (GeneRic Autonomic Signaling Protocol или GRASP). Такой интерфейс нужен агентам автономных служб (Autonomic Service Agent или ASA), вызывающим модуль протокола GRASP, для обмена сообщениями автономной сети (Autonomic Network) с другими агентами ASA. Поскольку протокол GRASP рассчитан на поддержку асинхронных операций, интерфейс API нужно приспосабливать к поддержке асинхронной работы в различных языках программирования и операционных системах.

Статус документа

Документ не содержит какой-либо спецификации (Internet Standards Track) и публикуется с информационными целями.

Документ является результатом работы IETF[1] и представляет согласованный взгляд сообщества IETF. Документ прошёл открытое обсуждение и был одобрен для публикации IESG[2]. Не все документы, одобренные IESG, претендуют на статус стандартов Internet, дополнительную информацию о стандартах можно найти в разделе 2 в RFC 7841.

Информацию о текущем статусе документа, ошибках и способах обратной связи можно найти по ссылке https://www.rfc-editor.org/info/rfc8991.

Оглавление

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

1. Введение

Как определено в [RFC8993], агент автономной службы (Autonomic Service Agent или ASA) - это неделимая (atomic) сущность автономной функции, создаваемая на самоуправляемых (autonomic) узлах. Эти узлы являются частью защищённой автономной плоскости управления (Autonomic Control Plane или ACP), такой, как определена в [RFC8994].

При взаимодействии между агентами ASA им следует применять базовый протокол автономной сигнализации (GeneRic Autonomic Signaling Protocol или GRASP) [RFC8990]. GRASP полагается на конфиденциальность и целостность, обеспечиваемые ACP, поэтому все узлы данной самоуправляемой сети (Autonomic Network) используют общую границу доверия, т. е. границу ACP. Узлы, не присоединившиеся к ACP, не могут передавать, принимать или перехватывать сообщения GRASP через ACP и не могут использовать адреса ACP. Агент ASA работает на узле ACP и при передаче сообщений через ACP использует свойства защиты, т. е., целостность и конфиденциальность сообщений, а также невозможность присоединения к ACP неуполномоченных узлов. В результате этого все ASA в данной самоуправляемой сети могут доверять сообщениям от других узлов. Поэтому определённый в документе интерфейс API не включает явных функций защиты.

Важным свойством GRASP является концепция задачи GRASP. Это структура данных, закодированная как и все сообщения GRASP в краткое представление бинарного объекта (Concise Binary Object Representation или CBOR) [RFC8949]. Основным содержимым структуры является имя и значение, как описано в разделе «Терминология» [RFC8990]. Когда задача передаётся от одного агента ASA к другому с использованием GRASP, значение переносится в одном направлении (синхронизация или лавинная отправка) или согласуется между двумя сторонами. Семантика значения не разбирается (opaque) GRASP и, следовательно, API. Каждая задача должна быть точно задана в выделенной спецификации, как указано в параграфе 2.10 «Опции задачи» [RFC8990]. В частности, спецификация задаёт синтаксис и семантику значения задачи, определяет поддержку согласования и режима пробного использования (dry-run), а также другие детали, требуемые для функциональной совместимости. Использование CBOR с языком краткого определения данных (Concise Data Definition Language или CDDL) [RFC8610] в качестве языка определения данных позволяет передавать значение между ASA независимо от применяемых языков программирования. За хранение и согласованность данных в процессе их согласования отвечают участвующие агенты ASA. Кроме того, протоколу GRASP требуется кэшировать последние значения задач, полученные лавинным путём.

На рисунке 1 показана реализация GRASP с несколькими подуровнями. Нижним уровнем является модуль базового протокола GRASP, который отвечает лишь за передачу и приём сообщений GRASP и поддержку общих структур данных. Над ним размещается базовый интерфейс API, описанный в этом документе. Верхний уровень содержит некоторые необязательные функции API, основанные на базовом протоколе GRASP. Например, в [GRASP-DISTRIB] описана одна из возможных функций расширения.

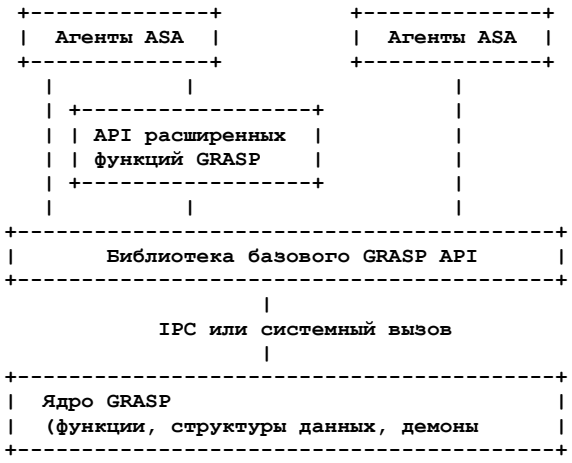

Рисунок 1. Программная схема.

Несколько ASA на одном узле будут совместно использовать один экземпляр GRASP, подобно использованию одного стека TCP/IP множеством приложений. Этот аспект скрыт от индивидуальных ASA интерфейсом API и далее не рассматривается.

Желательно создавать ASA в форме переносимых программ пользовательского пространства, применяющих системно-независимый API. Во многих реализациях код GRASP в результате разделён между пользовательским пространством и ядром. В пользовательском пространстве библиотечные функции обеспечивают API и напрямую взаимодействуют с ASA. В ядре демон или набор субслужб предоставляет не зависящие от конкретных ASA функции ядра GRASP, такие как обработка групповой передачи и ретрансляция, а также общие структуры данных, такие как кэш обнаружения. Библиотека GRASP API должна взаимодействовать с ядром GRASP через IPC[1] или системные вызовы. Детяли этих взаимодйствий зависят от системы.

Библиотеку GRASP и модули расширенных функций следует делать доступными для ASA. Однако расширенные функции могут добавляться постепенно, поэтому они могут быть описаны в будущих документах, а здесь описывается лишь базовый интерфейс GRASP API.

[1]Interprocess communication - взаимодействия между процессами.

Предоставляемые API функции не отображаются взаимно-однозначно на сообщения GRASP и предназначены, скорее, для удобной поддержки цепочек сообщений (таких как запрос обнаружения, за которым следуют отклики от нескольких партнёров, или запрос согласования, за которым следуют возможные отклики). Это сделано для того, чтобы помочь программистам ASA создавать код на основе требований приложений, не вникая в детали протокола.

Помимо компонентов автономной инфраструктуры, описанных в [RFC8994] и [RFC8995], простой самонастраиваемый узел может содержать лишь несколько ASA. Такой узел может напрямую включать стек GRASP в свой код и не требовать установки API. Однако в этом случае программистам придётся вникать в детали GRASP глубже, чем при работе через API.

Этот документ содержит концептуальное описание API и не является формальной спецификацией для какого-либо конкретного языка программирования или операционной системы. Уточнение деталей предполагается в реализациях.

2. GRASP API для ASA

2.1. Допущения при разработке

Предполагается, что агент ASA должен вызывать отдельную реализацию GRASP, которая обрабатывает протокольные детали (защита, отправка и прослушивание сообщений GRASP, ожидание, кэширование результатов обнаружения, циклы согласования, передача и приём при синхронизации данных и т. п.), но ничего не знает о конкретных задачах GRASP (см. параграф 2.10 в [RFC8990]). Семантика задач GRASP неизвестна протоколу и обрабатывается только агентами ASA. Это является абстрактным интерфейсом API для использования агентами ASA. Привязки к конкретным языкам следует задавать в отдельных документах.

Разные ASA могут применять функции GRASP по своему для:

- обнаружения;
- согласования в качестве инициатора (клиент);
- согласования в качестве ответчика;
- согласования в качестве инициатора или ответчика;
- синхронизации в качестве инициатора (получатель);
- синхронизации в качестве ответчика и/или выполняющего лавинную отправку;
- синхронизации в качестве инициатора, ответчика и/или выполняющего лавинную отправку.

API также предполагает возможность поддержки одним агентом ASA множества задач. Ничто не мешает ASA использовать одни задачи для синхронизации, а другие - для согласования.

Устройство API предполагает, что операционная система и язык программирования обеспечивают механизмы для одновременных асинхронных операций, как описано в параграфе 2.2.

Некоторые аспекты не включены в данную версию, поскольку они требуют практического опыта:

- предоставление полномочий агентам ASA не определено как часть GRASP и требует изучения;
- предоставляемые пользователем явные локаторы задач не поддерживаются; ядро GRASP предоставляет локатор, используя IP-адрес соответствующего узла;
- быстрый режим GRASP (параграф 2.5.4 в [RFC8990]) не поддерживается.

2.2. Асинхронные операции

GRASP зависит от асинхронных операций и состояний ожидания, а некоторые из сообщений протокола не являются идепотентными и повтор сообщения может повторно изменять состояние ASA-получателя. Многим ASA требуется поддержка нескольких одновременных операций, например, агенту ASA может потребоваться согласование одной задачи с партнёром, одновременно с обнаружением и синхронизацией другого объекта с другим партнёром. Кроме того, агенту ASA, служащему диспетчером ресурсов, может потребоваться одновременное согласование данной задачи с несколькими партнёрами. Вероятно, такому агенту ASA может потребоваться поддерживать неразрывные изменения во внутренних структурах данных с использованием механизмов, предоставляемых операционной системой и используемым языком программирования.

2.2.1. Варианты асинхронных механизмов

Некоторые ASA должны поддерживать асинхронные операции, поэтому ядро протокола GRASP должно делать это. В зависимости от операционной системы и применяемого языка программирования имеется несколько вариантов параллельной работы, три из которых рассмотрены ниже (многопоточность, циклы событий с использованием опроса, структура цикла событий с использованием обратных вызовов - callback).

1. При многопоточности операционная система и язык программирования обеспечивают требуемую поддержку асинхронных операций, включая создание новых потоков (thread), переключение контекста между потоками, очереди, блокировки и неявные состояния ожидания. В этом случае вызовы API можно считать просто синхронными вызовами в рамках потока, даже если функция включает состояния ожидания, блокировки и очередей. Одновременные операции выполняются в своих потоках. Например, вызов discover() может не возвращать управления, пока не поступит результат обнаружения или не возникнет тайм-аут. Если у ASA есть другая работа для выполнения, вызов discover() следует выполнять в отдельном потоке.
2. При реализации цикла событий с опросом блокирующие вызовы недопустимы, поэтому все вызовы должны быть неблокирующими, а главный цикл может поддерживать несколько сессий GRASP параллельно, периодически опрашивая каждую на предмет смены состояния. Для этого в реализации API предусматриваются неблокирующие версии всех функций, которые иначе могли бы включать блокировку и постановку в очередь. В этих вызовах будет возвращаться код noReply вместо блокировки, пока не произойдет

ожидаемое событие (или отказ). Например, вызов discover() будет возвращать noReply вместо ожидания обнаружения или тайм-аута. Вызов discover() будет повторяться в каждой итерации главного цикла, пока он не завершится. По сути это становится опросом.

3. Выше было отмечено, что некоторые сообщения GRASP не являются идемпотентными. В частности, это относится к каждому этапу сеансов согласования - отправка одного сообщения дважды может приводить к нежелательным побочным эффектам. Этого не возникает в опросе цикла событий - повторный вызов после получения noReply не повторяет сообщение, а просто проверяет наличие отклика.

4. В реализации цикла событий с обратными вызовами разработчик ASA создаёт callback-функцию для каждой асинхронной операции. Эта функция вызывается асинхронно при получении отклика или возникновении отказа (например, тайм-аута).

2.2.2. Несколько сессий согласования

Устройство GRASP позволяет реализовать описанный ниже сценарий. Рассмотрим ASA A, выступающий распределителем ресурсов для некой задачи. Агент ASA B запускает согласование с A для получения или освобождения некоторого объёма ресурса. Пока это согласование происходит, B решает запустить согласование с агентом A другого объёма того же ресурса. В результате агент A должен поддерживать два отдельных сеанса согласования, не смешивая их.

Отметим, что агенты ASA могут быть спроектированы так, чтобы избегать таких ситуаций, т. е. разрешать лишь 1 сеанс согласования в каждый момент для данной задачи, но это является произвольным ограничением, не требуемым протоколом GRASP. На деле GRASP предполагает, что для любого ASA, управляющего ресурсом, может потребоваться поддержка нескольких параллельный согласований, возможно, с одним и тем же партнёром. Схемы взаимодействия могут быть очень сложными, когда в группе агентов ASA согласования между собой перекрываются, как описано в [ANIMA-COORD]. Поэтому устройство API разрешает такие сценарии.

В модели с обратными вызовами для описанного выше сценария агенты A и B будут предоставлять два экземпляра callback-функции (по 1 для каждой сессии). По этой причине каждый агент ASA должен уметь различать сессии, но IP-адреса партнёра для этого недостаточно. Небезопасно и полагаться на номера транспортных портов, поскольку в будущих версиях GRASP могут применяться общие порты вместо отдельного порта для каждой сессии. Поэтому протокол GRASP включает Session ID. Таким образом, при необходимости в API различать одновременные сессии GRASP между парой партнёров используется дескриптор сессии (handle).

2.2.3. Перекрытие сессий и операций

Сессия GRASP состоит из конечной последовательности сообщений (для обнаружения, синхронизации или согласования) между парой ASA. Сессия однозначно указывается в линии псевдослучайным идентификатором Session ID и IP-адресом инициатора сессии. Дополнительные сведения приведены в параграфе 2.7 «Идентификатор сессии (Session ID)» [RFC8990].

При первом вызове в новой сессии GRASP интерфейс API возвращает дескриптор session_handle, однозначно указывающий сессию внутри API, что позволяет различать перекрывающиеся сеансы. Вероятная реализация дескриптора заключается в его формировании из базовых GRASP Session ID и адреса IP. Этот дескриптор должен применяться во всех последующих вызовах в рамках этой сессии (см. параграф 2.3.2.8).

Дополнительным механизмом повышения эффективности реализации опроса является добавление вызова общего назначений, скажем, notify(), который проверяет статус всех незавершённых операций для вызывающего агента ASA и возвращает значения session_handle для всех сеансов с изменившимся состоянием. Это избавит от необходимости повторных вызовов отдельных функций, возвращающих noReply. Этот вызов не описывается более подробно, поскольку его детали явно будут зависеть от реализации.

Следствием изложенного выше для всех реализаций GRASP является необходимость хранения ядром GRASP состояния каждой выполняемой операции GRASP, наиболее вероятно, указываемое GRASP Session ID и адресом инициатора сессии. Даже в потоковой реализации ядру GRASP требуются такие состояния для внутренней работы. Параметр session_handle раскрывает этот аспект реализации.

2.2.4. Завершение сессии

Сессии GRASP могут завершаться по разным причинам. Сеанс завершается при успешном обнаружении или по тайм-ауту, при успешном согласовании или отказе, при доставке результатов синхронизации, по отказу отвечать со стороны удалённого узла до истечения времени ожидания, по завершению цикла (счётчик) или при ошибке сетевого сокета. Отметим, что тайм-аут на одной стороне сессии может вызывать тайм-аут или ошибку сокета на другой стороне, поскольку GRASP в таких случаях не передаёт сообщения об ошибке. В любом случае API возвращает вызывающему подходящий код и тому следует освободить все выделенные для сессии ресурсы. При отказах спецификация GRASP рекомендует экспоненциально увеличивать интервал между повторами.

2.3. Определение API

2.3.1. Обзор функций

Предоставляемые API делятся на несколько групп, перечисленных ниже.

Registration - регистрация
: Эти функции позволяют ASA регистрироваться в ядре GRASP, а также позволяют зарегистрированным ASA регистрировать задачи GRASP, которыми они будут манипулировать.

Discovery - обнаружение
: Эта функция позволяет агенту ASA, которому нужно инициировать согласование или синхронизацию для конкретной задачи, обнаружить партнёра, готового ответить на запрос.

Negotiation - согласование

Эти функции позволяют ASA выступать инициатором (запрашивающий) или ответчиком (слушающий) для сессии согласования GRASP. После инициирования согласование является симметричным процессом и большинство функций может использовать любая из сторон.

Synchronization - синхронизация

Эти функции позволяют ASA выступать инициатором (запрашивающий) или ответчиком (слушающий) для сессии синхронизации GRASP.

Flooding - лавинная отправка

Эти функции позволяют ASA передавать и принимать задачу, рассылаемую лавинно всем узлам ACP.

В [ASA-GUIDE] представлены примеры логических потоков для управляющего ресурсами агента ASA, которые могут помочь разобраться с приведёнными ниже описаниями. В следующем параграфе описаны параметры и структуры данных, применяемые для вызовов API, а затем рассматриваются различные группы функций API. API, для которых не указаны асинхронные механизмы, неявно являются синхронными по своему поведению.

2.3.2. Параметры и структуры данных

2.3.2.1. Целые числа

В этом API целочисленные значения предполагаются 32-битовыми числами без знака (uint32_t), если не указано иное.

2.3.2.2. Коды ошибок

Все функции API имеют целое число errorcode в качестве кода возврата (первое возвращаемое значение в языках, позволяющих возвращать несколько значений). Значение errorcode = 0 указывает успешное выполнение, все остальные - ошибку. Три первых кода ошибок имеют особое значение, как указано ниже.

1 - Declined (отклонено)

Другая сторона передала сообщение GRASP Negotiation End (M_END) с опцией Decline (O_DECLINE).

2 - No reply (нет ответа)

Применяется в неблокируемых вызовах для указания того, что другая сторона пока не ответила (см. параграф 2.2).

3 - Unspecified error (неуказанная ошибка)

Используется в случаях отсутствия подходящего кода ошибки.

В Приложении A приведён полый список определённый в настоящее время кодов ошибок (на основе опыта реализации). От реализаций не требуется использовать одни и те же коды ошибок, но это настоятельно рекомендуется для обеспечения переносимости приложений.

2.3.2.3. Тайм-аут

Значения параметра timeout указываются в миллисекундах целым числом без знака. При нулевом значении параметра применяется заданный по умолчанию тайм-аут GRASP (GRASP_DEF_TIMEOUT [RFC8990]). Исключением является функция discover(), для которой нулевой тайм-аут интерпретируется иначе. Если в интервале timeout не получено ответа, вызов считается завершившимся отказом (когда явно не указано иное).

2.3.2.4. Задача

Параметр objective является структурой данных с описанными ниже полями.

name (UTF-8 string)

Имя задачи.

neg (Boolean flag)

True, если задача поддерживает согласование (по умолчанию False)

synch (Boolean flag)

True, если задача поддерживает синхронизацию (по умолчанию False)

dry (Boolean flag)

True, если задача поддерживает пробное (dry-run) согласование (по умолчанию False)

Примечание 1. Значение True может иметь лишь один из флагов synch и neg.

Примечание 2. Флаг Key не может иметь значения True, пока не задано neg = True.

Примечание 3. В некоторых языках программирования может быть предпочтительно указание флагов Boolean битами одного байта, который затем кодируется в сообщения GRASP. В других языках может быть предпочтительным перечисляемое значение (enumeration).

loop_count (целое число без знака, uint8_t)

Предельное число шагов согласования и т. п. (по умолчанию GRASP_DEF_LOOPCT [RFC8990]). Для loop_count подходящее значение задаёт инициатор согласования, чтобы предотвратить бесконечный цикл. Поле также служит для ограничения области распространения сообщений обнаружения и лавинной рассылки.

value

Конкретная структура данных, указывающая значение задачи. Структура зависит от языка программирования с учётом ограничений CBOR [RFC8949].

Важным преимуществом CBOR является то, что значение задачи может быть совершенно непонятным (opaque) для ядра GRASP, но при этом должно передаваться в агент ASA и из него. Хотя ядро GRASP должно проверять формат и синтаксис сообщений GRASP, оно не может проверять значение задачи и способно проверить лишь корректность формата CBOR. Обработка при декодировании зависит от применяемой библиотеки CBOR, но соответствующий код ошибки (CBORfail) задаётся в API и возвращается агенту ASA, если ошибочное сообщение может быть отнесено к текущей сессии GRASP. Однако каждый агент ASA отвечает за проверку значения полученной задачи, как указано в параграфе 5.3 [RFC8949]. Если используется подходящий объектно-ориентированный язык программирования, GRASP API может десериализовать значение и представить его ASA как задачу. В иных случаях значение представляется как объект данных CBOR. В любом случае синтаксис и семантика значения задачи относятся к зоне ответственности ASA.

Требованием ко всем привязкам к языку и реализациям API является возможность использования в качестве значения необработанного элемента данных CBOR, независимо от наличия других вариантов представления. API «оборачивает» его с тегом CBOR Tag 24 как элемент данных в кодировке CBOR для передачи через GRASP и

«разворачивает» при получении. Это обеспечивает агентам ASA возможность взаимодействия, не зависящего от языка программирования.

Поля name и value имеют переменный размер. GRASP не ограничивает размер этих полей, устанавливая лишь максимальный размер сообщения GRASP. Размер этих полей может ограничивать реализация.

Ниже представлен пример определения структуры данных для задачи на языке C с использованием версии не ниже C99 в предположении использования библиотеки CBOR [libcbor].

```
typedef struct {
   unsigned char *name;
   uint8_t flags;            // Биты флагов, определённые GRASP
   uint8_t loop_count;
   uint32_t value_size;      // Размер значения в байтах
   cbor_mutable_data cbor_value;
                             // CBOR bytestring (libcbor/cbor/data.h)
               } objective;
```

Ниже представлен пример определения структуры данных для задачи на языке Python (версия 3.4 или выше).

```
class objective:
   """A GRASP objective"""
   def __init__(self, name):
       self.name = name         #Уникальное имя (string)
       self.negotiate = False   #True, если поддерживается согласование
       self.dryrun = False      #True, если поддерживается dry-run
       self.synch = False       #True, если поддерживается синхронизация
       self.loop_count = GRASP_DEF_LOOPCT  # Принятое по умолчанию стартовое значение
       self.value = None        #Владелец (любой объект Python)
```

2.3.2.5. asa_locator

Параметр asa_locator является структурой данных с описанными ниже полями.

locator
Фактический локатор (адрес IP или строка ASCII).

ifi (целое число без знака)
Индекс идентификатора интерфейса, через который происходило обнаружение (ограничено в большинстве ASA).

expire (тип зависит от системы)
Время по локальным часам, когда локатор будет исключён из кэша.

Ниже указаны все типы локаторов, поддерживаемые GRASP в настоящее время.

is_ipaddress (Boolean)
True, если локатором является адрес IP.

is_fqdn (Boolean)
True, если локатором является полное доменное имя (Fully Qualified Domain Name или FQDN).

is_uri (Boolean)
True, если локатором является URI.

Эти варианты являются взаимоисключающими. В зависимости от языка программирования они могут быть представлены битовыми шаблонами или перечисляемыми значениями.

diverted (Boolean)
True, если локатор обнаружен с помощью опции Divert.

protocol (целое число без знака)
Применяемый транспортный протокол (IPPROTO_TCP или IPPROTO_UDP). Эти константы заданы в CDDL-спецификации GRASP [RFC8990].

port (целое число без знака)
Применяемый номер порта.

Поле locator имеет переменный размер в случае FQDN или URI. GRASP не ограничивает размер поля, устанавливая лишь максимальный размер сообщения GRASP, но реализация может вносить свои ограничения.

Следует отметить, что при обнаружении ASA значения asa_locator другого агента не применяется явной проверки подлинности. В соответствии с моделью доверия защищённой плоскости ACP агенты ASA предполагаются предоставляющими при обнаружении корректные локаторы. Дополнительные сведения приведены в параграфе 2.9.5 «Опции локации» [RFC8990].

2.3.2.6. tagged_objective

Параметр tagged_objective является структурой данных с описанными ниже полями.

objective
Задача.

locator
Связанный с задачей параметр asa_locator или значение null.

2.3.2.7. asa_handle

Хотя схемы проверки подлинности и полномочий для ASA не определены, API обеспечивает очень простую ловушку для такой схемы. При старте ASA агент регистрируется в ядре GRASP, которое предоставляет ему неинтерпретируемый (opaque) дескриптор, не имеющий криптографической защиты, но сложный для предсказания посторонними. Агент ASA должен предоставлять этот дескриптор при последующих вызовах. Этот механизм предотвращает некоторые элементарные ошибки и тривиальные атаки, такие как манипуляции с задачами со стороны незарегистрированных ASA.

Таким образом, параметр asa_handle требуется для большинства вызовов. Он создаётся при регистрации ASA в GRASP и агент ASA должен сохранять asa_handle и применять его при каждом последующем вызове GRASP. Вызовы с непригодным дескриптором будут завершаться отказом. Дескриптор является 32-битовым неинтерпретируемым значением (например, uint32_t). Поскольку дескриптор применяется лишь локально без передачи в сообщениях

GRASP, его уникальность должна обеспечиваться лишь в рамках экземпляра GRASP. Дескриптор действует до завершения работы ASA. Дескриптор следует делать непредсказуемым, например, с использованием того же механизма, который применяется GRASP для создания Session ID (параграф 2.3.2.8).

2.3.2.8. session_handle и обратные вызовы

В некоторых вызовах требуется параметр session_handle, который является структурой данных, не интерпретируемой (opaque) ASA, и служит для индикации вызовов API, связанных с конкретной сессией GRASP (см. параграф 2.2.3). Параметр применяется в обратных вызовах (callback). Кроме указания конкретной сессии этот параметр позволяет GRASP обнаруживать и игнорировать вызовы из несуществующих и просроченных сессий.

В реализациях с циклом событий (параграф 2.2.1) обратные вызовы могут поддерживаться для всех функций API, предполагающих ожидание для удалённой операции:

discover() с обратным вызовом discovery_received();

request_negotiate() с обратным вызовом negotiate_step_received();

negotiate_step() с обратным вызовом negotiate_step_received();

listen_negotiate() с обратным вызовом negotiate_step_received();

synchronize() с обратным вызовом synchronization_received().

Детали обратных вызовов зависят от реализации.

2.3.3. Регистрация

Эти функции служат для регистрации ASA и изменяемых агентом задач в модуле GRASP. При отсутствии модели проверки полномочий эти функции очень просты, но позволяют избежать выбора одного имени несколькими ASA и манипуляций с одной задачей нескольких агентов ASA. При включении в GRASP модели проверки полномочий эти вызовы API потребуется соответственно изменить.

register_asa()

Все агенты ASA должны использовать этот вызов до обращения к каким-либо иным функциям API.

- Входной параметр:
  имя (name) ASA (UTF-8 string).
- Возвращаемое значение:
  errorcode (целое число без знака)
  asa_handle (целое число без знака)
- Функция инициализирует в модуле GRASP состояние для вызывающего элемента (ASA). При успешном выполнении функция возвращает дескриптор asa_handle, который агент ASA должен предоставлять в последующих вызовах. В случае отказа ASA не получает полномочий и не может работать. Значение asa_handle будет неопределённым.

deregister_asa()

- Входные параметры:
  asa_handle (целое число без знака);
  имя (name) ASA (UTF-8 string).
- Возвращаемое значение:
  errorcode (целое число без знака).
- Удаляет из модуля GRASP все состояния для вызывающего элемента (ASA) и отменяет регистрацию всех зарегистрированных им задач. Эти действия должны выполняться автоматически при завершении работы ASA.
- Имя ASA, строго говоря, является избыточным в таких вызовах, но сохранено для обнаружения и отклонения ошибочных отмен регистрации.

register_objective()

Агенты ASA должны использовать этот вызов для любой задачи, которая передаётся при согласовании, синхронизации или лавинной отправке.

- Входные параметры:
  asa_handle (целое число без знака);
  objective (structure);
  ttl (целое число без знака, по умолчанию GRASP_DEF_TIMEOUT);
  discoverable (Boolean, по умолчанию False);
  overlap (Boolean, по умолчанию False);
  local (Boolean, по умолчанию False).
- Возвращаемое значение:
  errorcode (целое число без знака).
- Функция регистрирует задачу, которую этот агент ASA может изменять и передавать другим ASA при лавинной рассылке или согласовании. Не требуется регистрировать задачи, которые лишь принимаются GRASP при синхронизации или лавинной рассылке. Задача (objective) становится кандидатом для обнаружения. Однако отклики на обнаружение не следует разрешать, пока ASA не вызовет функцию listen_negotiate() или listen_synchronize(), показывающую способность агента выступать ответчиком. ASA может согласовывать задачу, передавать данные синхронизации или рассылать данные лавинно. Регистрация не требуется для операций лишь чтения (read-only), когда ASA хочет лишь принимать данные синхронизации или лавинной рассылки для соответствующей задачи.
- Параметр ttl указывает действительный срок действия (время жизни) в миллисекундах любого отклика на обнаружение, созданного для этой задачи. По умолчанию следует применять заданный по умолчанию тайм-аут GRASP (GRASP_DEF_TIMEOUT, см. [RFC8990]).
- Если discoverable = True, задача сразу же становится доступной для обнаружения. Это предназначено для задач, которые определены лишь для обнаружения GRASP и не поддерживают согласование и синхронизацию.

- Если overlap = True, эта задача может регистрироваться несколькими ASA в одном экземпляре GRASP. Это имеет значение для управления жизненным циклом агентов ASA [ASA-GUIDE] и должно применяться согласованно для данной задачи (всегда True или всегда False).
- Если local = True, обнаружение должно возвращать адрес локального канала (link-local). Это предназначено для задач, которые должны быть ограничены локальным каналом.
- Этот вызов может повторяться для нескольких задач.

deregister_objective()

- Входные параметры:
  asa_handle (целое число без знака);
  objective (structure).
- Возвращаемое значение:
  errorcode (целое число без знака).
- Задача objective должна быть зарегистрирована вызовом ASA, в противном случае этот вызов ведёт к отказу. При дерегистрации удаляются все состояния в модулей GRASP для данной задачи.

2.3.4. Обнаружение

discover()

Эту функцию может применять любой агент ASA для обнаружения партнёров, обрабатывающих данную задачу.

- Входные параметры:
  asa_handle (целое число без знака);
  objective (structure);
  timeout (целое число без знака);
  minimum_TTL (целое число без знака).
- Возвращаемые значения:
  errorcode (целое число без знака);
  locator_list (structure).
- Функция возвращает список asa_locator для данной задачи, пустой список говорит о том, что за время ожидания (timeout) локаторов не найдено. Структура включает поля, указанные в параграфе 2.3.2.5.
- Параметр minimum_TTL должен быть не меньше 0. Все кэшированные локаторы для задачи, срок действия которых в миллисекундах не превышает значения minimum_TTL, удаляются, поэтому при значении minimum_TTL = 0 исключаются все записи. Отметим, что это не влияет на продолжающиеся сессии, которые используют удаляемые локаторы.
- Если timeout = 0, сразу же возвращаются все остающиеся локально кэшированные локаторы для задачи и других действий не выполняется. Поэтому вызов с minimum_TTL = 0 и timeout = 0 не имеет смысла.
- Если timeout > 0, выполняется обнаружение GRASP и возвращаются все результаты, полученные до завершения времени ожидания (timeout). Если результатов нет, после тайм-аута возвращается пустой список и это не считается ошибкой. Обнаружение GRASP не является детерминированным процессом. Если имеется несколько узлов, обрабатывающих задачу до истечения тайм-аута могут быть обнаружены все, часть или ни одного из таких узлов.
- Асинхронные механизмы
  Потоковая реализация
  Если нужна асинхронная операцию, эту функцию следует вызывать в отдельном потоке.
  Реализация цикла событий
  Используется дополнительный параметр session_handle. Если errorcode имеет значение 2 (noReply), это говорит, что отклик ещё не получен. Параметр session_handle должен указываться при последующих вызовах. При ненулевом времени ожидания можно использовать обратный вызов (callback).

2.3.5. Согласование

Механизм согласования отличается от типичного обмена между клиентом и сервером, поэтому на рисунке 2 показана последовательность вызовов и сообщений GRASP при согласовании. Отметим, что после первого протокольного обмена процесс становится симметричным и шаги согласования строго чередуются между двумя сторонами. Согласование может завершить любая из сторон. Кроме того, сторона, которой нужно отвечать следующей, может в любой момент внести задержку, для продления времени ожидания другой стороны. Это может применяться, например, если ASA требуется согласование с третьей стороной для продолжения текущего согласования.

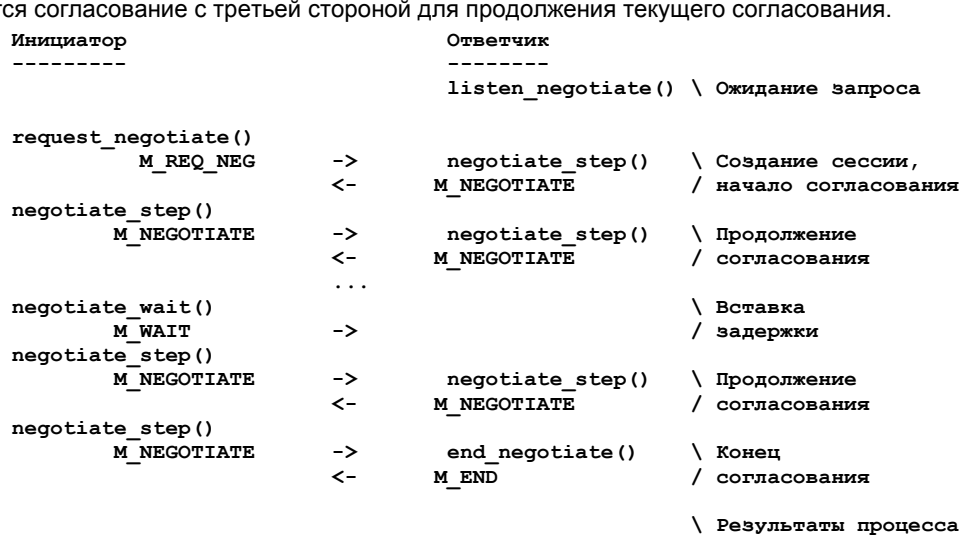

Рисунок 2. Последовательность согласования.

Счётчик циклов, встроенный в задачу, являющийся предметом согласования, инициализируется агентом ASA, начинающим согласование, и декрементируется ядром GRASP на каждом шаге перед отправкой сообщения M_NEGOTIATE. При достижении 0 согласование завершается отказом и каждая сторона получает код ошибки.

В процессе согласования каждая сторона обновляет значение задачи в соответствии с её семантикой на основе спецификации задачи. Хотя у многих задач значения могут быть упорядочены, чтобы согласование было простым процессом «торгов», это не является требованием.

Отказ при согласовании, тайм-аут или обнуление счётчика цикла могут завершать сеанс согласования, но это не считается отказом протокола.

request_negotiate()

Эта функция применяется агентом ASA для инициирования согласования задачи GRASP в качестве запрашивающей стороны (клиента).

- Входные параметры:
  asa_handle (целое число без знака);
  objective (structure);
  peer (asa_locator);
  timeout (целое число без знака).
- Возвращаемые значения:
  errorcode (целое число без знака);
  session_handle (structure) (не определена до успешного завершения);
  proffered_objective (structure) (не определена до успешного завершения);
  reason (string) (пуста, пока согласование не отвергнуто).
- Функция открывает сеанс согласования между парой ASA. Отметим, что GRASP с настоящее время не поддерживает многосторонние согласования, которые требуется добавить в виде расширенной функции.
- Параметр objective должен включать запрашиваемое значение, а для его счётчика цикла агенту ASA следует устанавливать подходящее значение. В противном случае будет использоваться принятое по умолчанию значение GRASP.
- Отметим, что данный сеанс согласования может (но не обязан) быть пробным (dry-run); два режима недопустимо смешивать в одной сессии.
- Параметр peer указывает целевой узел, это должно быть значение asa_locator, возвращённое функцией discover(). При пустом значении peer сначала автоматически выполняется обнаружение GRASP для нахождения подходящего партнёра (т. е., любого узла, поддерживающего рассматриваемую задачу).
- Параметр timeout описан в параграфе 2.3.2.3.
- Если в errorcode возвращается значение 0, это говорит об успешном начале согласования. Здесь возможны два случая.
  1. Пустой параметр session_handle. Согласование завершилось за 1 обмен и партнёр воспринял запрос. Возвращённая структура proffered_objective содержит значение, воспринятое партнёром, которое, следовательно, совпадает с objective в запросе. Дескриптор сессии не нужен, так как она завершена.
  2. Непустой параметр session_handle. Согласование должно продолжаться и в последующих шагах должно указываться значение session_handle. Возвращённая структура proffered_objective содержит первое значение, предпочтённое партнёром обмене сообщениями, иными словами, встречное предложение. Содержимое этого экземпляра objective должно использоваться для подготовки следующего шага согласования (см. negotiate_step() ниже), поскольку в нем указан счётчик цикла от партнёра. Код GRASP автоматически декрементирует счётчик на 1 при каждом шаге и возвращает ошибку при достижении 0. Поскольку обнуление счётчика прерывает согласование, на другой стороне возникает тайм-аут, прерывающий сессию.

  За этой функцией должны следовать вызовы negotiate_step, negotiate_wait, end_negotiate, пока согласование не завершится. После этого можно вызвать request_negotiate для нового согласования.
- Значение errorcode = 1 (declined) указывает, что партнёр отверг согласование (M_END и O_DECLINE в GRASP). Строка reason содержит сведения о причине отказа (для информирования и диагностики), но может быть пустой. При этой и других ошибках рекомендуется экспоненциально увеличивать интервал до следующего вызова (см. раздел 3).
- Асинхронные механизмы
  Потоковая реализация
  Для асинхронных операций функцию следует вызывать в отдельном потоке.
  Реализация цикла событий
  Параметр session_handle служит для различения одновременных сессий. Значение errorcode = 2 (noReply) указывает, что отклик ещё не получен. В последующих вызовах должен применяться параметр session_handle.
- Использование режима dry-run должно быть согласовано с сессией GRASP. Состояние флага Key в исходном вызове request_negotiate() должно сохраняться при последующих вызовах в той же сессии. Семантика режима пробного запуска (dry-run) встроена в ASA, а GRASP лишь передаёт бит флага.
- Для агентов ASA инфраструктуры ACP вполне вероятна необходимость обнаружения и согласования с партнёрами в каждом из соседей по каналу связи (on-link). Поэтому агенту нужно знать не только адрес link-local IP, но также физический интерфейс и транспортный порт для каждого соседа. Одним из вариантов реализации этого является включение таких деталей в структуру данных session_handle, не интерпретируемую (opaque) обычными ASA.

listen_negotiate()

Эту функцию агент ASA использует для начала своей работы в качестве ответчика при согласовании (слушающий) для данной задачи GRASP.

- Входные параметры:
  asa_handle (целое число без знака);
  objective (structure).
- Возвращаемые значения:
  errorcode (целое число без знака);
  session_handle (structure) (не определена до успешного завершения);
  requested_objective (structure) (не определена до успешного завершения).
- Эта функция указывает GRASP прослушивать запросы согласования для задачи objective, а также включает отклики на запросы обнаружения данной задачи, как указано в описании register_objective() (параграф 2.3.3).

- Асинхронные механизмы
  Потоковая реализация
  Функция блокируется в состоянии ожидания входящего запроса, поэтому её следует вызывать в отдельном потоке, если нужна асинхронная операция. Если не возникло неожиданного отказа, вызов возвращает управление только после получения входящего запроса на согласование. Если ASA поддерживает несколько одновременных транзакций, для каждой новой сессии должен запускаться новый субпоток, чтобы можно было сразу вызвать listen_negotiate().
  Реализация цикла событий
  Для различения сессий применяется параметр session_handle. Если ASA поддерживает несколько одновременных транзакций, для каждой новой сессии должен внедряться новый цикл событий, чтобы можно было сразу вызвать listen_negotiate().
- Этот вызов возвращает управление (потоковая модель) или вызывает срабатывание триггера (цик событий) лишь после получения входящего запроса. Когда это происходит, requested_objective содержит первое значение, запрашиваемое партнёром. Содержимое этого экземпляра задачи (objective) должно использоваться при последующих вызовах, поскольку в нем указан счётчик цикла от партнёра по согласованию. Параметр session_handle должен указываться во всех последующих шагах согласования.
- За этой функцией должен следовать вызов negotiate_step, negotiate_wait, end_negotiate, пока согласование не завершится.
- Если агент ASA способен одновременно обслуживать несколько согласований, он может вызвать listen_negotiate в нескольких потоках одновременно или вставить несколько событий. API и реализация GRASP должны поддерживать реентерабельное использование статуса прослушивания и вызовов согласования. Одновременные сессии будут различаться по самим потокам или событиям, дескрипторам сессий GRASP и базовыми транспортными сокетами (индивидуальными - unicast).

stop_listen_negotiate()

Эту функцию ASA вызывает для остановки работы в качестве ответчика (слушающего) для данной задачи GRASP.

- Входные параметры:
  asa_handle (целое число без знака);
  objective (structure).
- Возвращаемое значение:
  errorcode (целое число без знака).
- Указывает GRASP прекратить прослушивание запросов на согласование для данной задачи, т. е. отменить listen_negotiate().
- Асинхронные механизмы
  Потоковая реализация
  Должна вызваться из другого потока, нежели listen_negotiate().
  Реализация цикла событий
  Нет специальных условий.

negotiate_step()

Эту функцию ASA применяет в сеансе согласования для выполнения следующего шага.

- Входные параметры:
  asa_handle (целое число без знака);
  session_handle (structure);
  objective (structure);
  timeout (целое число без знака), как описано в параграфе 2.3.2.3.
- Возвращаемые значения:
  Как для вызова request_negotiate().
- Выполняет следующий шаг согласования с партнёром. Параметр objective содержит следующее значение, предлагаемое ASA на этом шаге, а также должен включать последнее значение loop_count, полученное при вызове request_negotiate() или negotiate_step().
- Асинхронные механизмы
  Потоковая реализация
  Обычно функция вызывается в одном потоке с предшествующим request_negotiate или listen_negotiate с тем же значением session_handle.
  Реализация цикла событий
  Должно использоваться значение session_handle, возвращённое предыдущим вызовом request_negotiate() или listen_negotiate().

negotiate_wait()

Эту функцию вызывает любой из участвующих в согласовании агентов ASA для задержки следующего шага.

- Входные параметры:
  asa_handle (целое число без знака);
  session_handle (structure);
  timeout (целое число без знака).
- Возвращаемое значение:
  errorcode (целое число без знака).
- Запрос к удалённому партнёру задержать сеанс согласования на timeout миллисекунд сверх исходного времени ожидания. Эта функция просто вызывает отправку сообщения GRASP Confirm Waiting [RFC8990].
- Асинхронные механизмы
  Потоковая реализация
  Вызывается в одном потоке с предшествующим request_negotiate() или listen_negotiate() с тем же session_handle.
  Реализация цикла событий
  Должно использоваться значение session_handle, полученное из предыдущего вызова request_negotiate() или listen_negotiate().

end_negotiate()

Эту функцию использует любой из участвующих в согласовании агентов ASA для завершения сессии.

- Входные параметры:
  asa_handle (целое число без знака);

session_handle (structure);
result (Boolean);
reason (UTF-8 string);
- Возвращаемое значение:
errorcode (целое число без знака).
- Завершение сеанса согласования:
result = True при успешном согласовании, False при отказе (отклонении);
reason = строка, указывающая причину отклонения (может быть пустой, при успехе игнорируется).
- Асинхронные механизмы
Потоковая реализация
Вызывается в одном потоке с предшествующим request_negotiate() или listen_negotiate() с тем же session_handle.
Реализация цикла событий
Должно использоваться значение session_handle, полученное из предыдущего вызова request_negotiate() или listen_negotiate().

2.3.6. Синхронизация и лавинная рассылка

synchronize()

Эту функцию агент ASA использует для запуска синхронизацию задачи GRASP, будучи ответчиком (клиентом).

- Входные параметры:
asa_handle (целое число без знака);
objective (structure);
peer (asa_locator);
timeout (целое число без знака).
- Возвращаемые значения:
errorcode (целое число без знака);
result (structure) (не определена до успешного завершения).
- Этот вызов запрашивает синхронизированное значение данной задачи (objective).
- Если параметр peer пуст и задача уже доступна в локальном кэше, рассылаемая лавинно задача возвращается сразу в параметре result. Значение timeout в этом случе игнорируется.
- Если параметр peer непуст или задача недоступна в локальном кэше, выполняется синхронизация с найденным агентом ASA. При успешном выполнении найденная задача возвращается в параметре result.
- Параметр peer содержит значение asa_locator, возвращённое функцией discover(). При пустом peer сначала автоматически выполняется обнаружение GRASP для поиска подходящего партнёра (любого узла, поддерживающего данную задачу).
- Параметр timeout описан в параграфе 2.3.2.3.
- Этот вызов следует повторяит всякий раз, когда требуется последнее значение.
- Асинхронные механизмы
Потоковая реализация
Вызов выполняется в отдельном потоке, если нужна асинхронная операция.
Реализация цикла событий
Используется дополнительный входной/выходной параметр session_handle как для request_negotiate(). Значение errorcode = 2 (noReply) говорит, что отклик ещё не получен. При последующих вызовах должен указываться параметр session_handle.
- При отказе перед повтором следует использовать экспоненциально увеличиваемый интервал (раздел 3).

listen_synchronize()

Эту функцию агент ASA использует для начала работы в качестве ответчика (слушающего) при синхронизации для данной задачи GRASP.

- Входные параметры:
asa_handle (целое число без знака);
objective (structure).
- Возвращаемое значение:
errorcode (целое число без знака).
- Эта функция указывает GRASP прослушивать запросы синхронизации для данной задачи и отвечать значением, заданным в параметре objective, а также включает отклики для задачи, как указано в описании register_objective() (параграф 2.3.3).
- Этот вызов является неблокирующим и может повторяться при изменении значения.

stop_listen_synchronize()

Эту функцию агент ASA использует для прекращения работы в качестве ответчика (слушающего) при синхронизации для данной задачи GRASP.

- Входные параметры:
asa_handle (целое число без знака);
objective (structure).
- Возвращаемое значение:
errorcode (целое число без знака).
- Эта функция указывает GRASP прекратить прослушивание запросов синхронизации для данной задачи objective, т. е. отменить предыдущий вызов listen_synchronize().

flood()

Эту функцию агент ASA применяет для лавинной рассылки одной или нескольких задач GRASP через автономную сеть. Отметим, что каждый узел GRASP кэширует все лавинно рассылаемые задачи, которые он получает, пока сроки их действия не завершится. Кэшированные задачи указываются с их источником и сроком действия, поэтому одновременно может кэшироваться несколько копий одной задачи. Подробности приведены в параграфе 2.8.11 «Сообщение Flood Synchronization» [RFC8990].

- Входные параметры:
asa_handle (целое число без знака);
ttl (целое число без знака);
tagged_objective_list (structure).

- Возвращаемое значение:
  errorcode (целое число без знака).
- Этот вызов указывает GRASP выполнить лавинную рассылку указанных целей для синхронизации, из значений и связанных локаторов всем узлам GRASP.
- Параметр ttl указывает срок действия рассылаемых лавинно данных в миллисекундах (0 указывает неограниченное действие).
- Параметр tagged_objective_list содержит 1 или несколько пар tagged_objective. Параметр locator для каждой задачи обычно пуст, но может быть действительным asa_locator. Инфраструктурные ASA, которым нужно лавинно разослать триблет {address, protocol, port} с задачей, создают для этого объект asa_locator. Если в качестве IP-адреса в таком локаторе применяется незаданный адрес (::), он заменяется адресом link-local передающего узла в каждой копии группового лавинного сообщения с принудительно заданным счётчиком цикла 1. Это предназначено для задач, ограниченных локальным каналом.
- Функция проверяет, что агент ASA зарегистрировал каждую задачу.
- Вызов может повторяться при изменении любого значения.

get_flood()

Эту функцию агент ASA применяет для получения текущего значения лавинно разосланной задачи GRASP.

- Входные параметры:
  asa_handle (целое число без знака);
  objective (structure).
- Возвращаемые значения:
  errorcode (целое число без знака);
  tagged_objective_list (structure) (не определена до успешного завершения).
- Этот вызов указывает GRASP возвратить данную задачу для синхронизации, если она разослана лавинно и срок её действия не истёк.
- Параметр tagged_objective_list содержит список пар tagged_objective, каждая из которых является копией разосланной лавинно задачи и соответствующего локатора. Такие образом, если одну задачу лавинно рассылали несколько ASA, получатель может различить копии.
- Этот вызов предназначен для смежных ASA. В простом случае ASA может просто вызвать synchronize() для получения действительной лавинно разосланной задачи.

expire_flood()

Эту функцию агент ASA может применять для завершения срока действия конкретных записей в локальном кэше лавинной рассылки GRASP.

- Входные параметры:
  asa_handle (целое число без знака);
  tagged_objective (structure).
- Возвращаемое значение:
  errorcode (целое число без знака).
- Вызов можно применять лишь после предшествующего get_flood() агентом ASA способным понять, что рассылаемое лавинно значение устарело или недействительно. Функцию следует применять осторожно.
- Параметр tagged_objective указывает устаревший элемент.

2.3.7. Функция для недействительных сообщений

send_invalid()

Эту функцию может использовать любой агент ASA для остановки продолжающейся сессии GRASP.

- Входные параметры:
  asa_handle (целое число без знака);
  session_handle (structure);
  info (байты).
- Возвращаемое значение:
  errorcode (целое число без знака).
- Функция передаёт сообщение GRASP Invalid (M_INVALID), как описано в [RFC8990]. Её не следует применять, если достаточно будет вызова end_negotiate(). Отметим, что сообщение может использоваться в ответ на любое индивидуальное сообщение GRASP, которое получатель не может корректно интерпретировать. В большинстве случаев сообщение будет создаваться внутри реализации GRASP. Параметр info может содержать необязательные диагностические сведения от ASA. Это могут быть необработанные байты из непонятного сообщения.

3. Вопросы безопасности

Вопросы безопасности для протокола GRASP рассмотрены в [RFC8990], включая атаки с отказом в обслуживании, хотя для ACP они не считаются сильно опасными. В различных местах протокола GRASP рекомендуется применять экспоненциальное повышение интервалов (backoff). Агенту ASA, применяющему API, следует использовать экспоненциальное увеличение интервала после неудачных вызовов discover(), req_negotiate(), synchronize(). Масштаб интервалов для таких операций зависит от семантики соответствующей задачи GRASP. Вызов flood() не следует повторять через более короткие интервалы, чем это будет полезно (в зависимости от семантики соответствующей задачи GRASP). Эти предостережения призваны помочь при обнаружении атак на службы.

В качестве общей предосторожности все ASA, способные обрабатывать параллельно несколько запросов согласования или регистрации, могут защититься от атак на службы, ограничивая число обрабатываемых одновременно запросов и отбрасывая без уведомления избыточные запросы. Для ядра GRASP может быть полезно ограничивать число задач, регистрируемых данным агентом ASA, общее числе регистрируемых всеми ASA задач и общее число одновременных сессий для защиты ресурсов системы. При высокой автономной активности, например, во время восстановления после обширных повреждений агенты ASA могут сталкиваться с частыми отказами сессий GRASP. Рекомендации по обеспечению устойчивости ASA приведены в [ASA-GUIDE].

Как уже отмечено, модель доверия предполагает, что все ASA в данной автономной сети взаимодействуют через защищённую автономную плоскость управления, поэтому могут доверять сообщениям друг от друга. Конкретная проверка полномочий ASA для использования определённых задач GRASP требует дополнительного изучения и кратко рассмотрена в [RFC8990].

Внимательный читатель отметит, что вредоносный агент ASA может расширить согласование на неопределённое время, используя функцию negotiate_wait() или меняя счётчик циклов в задаче. Отказоустойчивый агент ASA может обнаружить такое поведение партнёра и прервать согласование.

Параметр asa_handle применяется в API в качестве первой линии защиты от вредоносных программ, пытающихся имитировать корректно зарегистрированный агент ASA. Параметр session_handle служит в API первой линией защиты от враждебных процессов, пытающихся перехватить сессии GRASP. Оба эти дескриптора будут, скорей всего, создаваться с использованием 32-битового псевдослучайного GRASP Session ID. По своему устройству GRASP предотвращает риск конфликтов Session ID (см. параграф 2.7 «Идентификатор сессии (Session ID)» в [RFC8990]). Однако остаётся вероятность угадывания злоумышленником значения Session ID, session_handle или asa_handle, но это будет полезно лишь для атакующего, уже проникшего в ACP, которому доступны более простые формы атак, нежели перехват сессий GRASP.

4. Взаимодействие с IANA

Этот документ не требует действий IANA.

5. Литература

5.1. Нормативные документы

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/info/rfc8610>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/info/rfc8949>.

[RFC8990] Bormann, C., Carpenter, B., Ed., and B. Liu, Ed., "GeneRic Autonomic Signaling Protocol (GRASP)", RFC 8990, DOI 10.17487/RFC8990, May 2021, <https://www.rfc-editor.org/info/rfc8990>.

5.2. Дополнительная литература

[ANIMA-COORD] Ciavaglia, L. and P. Peloso, "Autonomic Functions Coordination", Work in Progress, Internet-Draft, draft-ciavaglia-anima-coordination-01, 21 March 2016, <https://tools.ietf.org/html/draft-ciavaglia-anima-coordination-01>.

[ASA-GUIDE] Carpenter, B., Ciavaglia, L., Jiang, S., and P. Peloso, "Guidelines for Autonomic Service Agents", Work in Progress[1], Internet-Draft, draft-ietf-anima-asa-guidelines-00, 14 November 2020, <https://tools.ietf.org/html/draft-ietf-anima-asa-guidelines-00>.

[GRASP-DISTRIB] Liu, B., Xiao, X., Hecker, A., Jiang, S., Despotovic, Z., and B. Carpenter, "Information Distribution over GRASP", Work in Progress, Internet-Draft, draft-ietf-anima-grasp-distribution-02, 8 March 2021, <https://tools.ietf.org/html/draft-ietf-anima-grasp-distribution-02>.

[libcbor] Kalvoda, P., "libcbor - libcbor 0.8.0 documentation", April 2021, <https://libcbor.readthedocs.io/>.

[RFC8993] Behringer, M., Ed., Carpenter, B., Eckert, T., Ciavaglia, L., and J. Nobre, "A Reference Model for Autonomic Networking", RFC 8993, DOI 10.17487/RFC8993, May 2021, <https://www.rfc-editor.org/info/rfc8993>.

[RFC8994] Eckert, T., Ed., Behringer, M., Ed., and S. Bjarnason, "ACK Autonomic Control Plane (ACP)", RFC 8994, DOI 10.17487/RFC8994, May 2021, <https://www.rfc-editor.org/info/rfc8994>.

[RFC8995] Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructure (BRSKI)", RFC 8995, DOI 10.17487/RFC8995, May 2021, <https://www.rfc-editor.org/info/rfc8995>.

Приложение A. Коды ошибок

В этом приложении указаны коды ошибок, определённые к настоящему времени на основе опыта реализации с символьными именами и краткими описаниями. Предполагается, что полные реализации API будут включать строки описаний на разных языках и могут включать дополнительные коды ошибок.

Коды ошибок, которые могут возвращать лишь одна или две функции, помечены именами этих функций в строке описания. Код noSecurity может возвращать большинство вызовов, если GRASP работает без защиты (например, нет защитной подложки, такой как ACP), за исключением специфического режима DULL, описанного в параграфе 2.5.2 «DULL GRASP» [RFC8990].

Таблица 1. Коды ошибок.

Имя	*Код*	*Описание*
ok	0	Все хорошо
declined	1	Отклонено (req_negotiate, negotiate_step)
noReply	2	Нет отклика (indicates waiting state in event loop calls)
unspec	3	Неуказанная ошибка
ASAfull	4	Реестр ASA заполнен (register_asa)
dupASA	5	Дублирование имени ASA (register_asa)
noASA	6	Агент ASA не зарегистрирован
notYourASA	7	Агент ASA зарегистрирован другим (deregister_asa)
notBoth	8	Задача не может поддерживать согласование и синхронизацию (register_obj)
notDry	9	Пробный прогон разрешён лишь при синхронизации (register_obj)

[1]Опубликовано в RFC 9222. *Прим. перев.*

notOverlap	10	Реализация не поддерживает наложение (register_obj)
objFull	11	Реестр задач заполнен (register_obj)
objReg	12	Задача уже зарегистрирована (register_obj)
notYourObj	13	Задача не зарегистрирована этим ASA
notObj	14	Задача не найдена
notNeg	15	Задача не согласуется (req_negotiate, listen_negotiate)
noSecurity	16	Нет защиты
noDiscReply	17	Нет отклика при обнаружении (req_negotiate)
sockErrNegRq	18	Ошибка сокета, передающего запрос согласования (req_negotiate)
noSession	19	Нет сессии
noSocket	20	Нет сокета
loopExhausted	21	Завершился счётчик цикла (negotiate_step)
sockErrNegStep	22	Ошибка сокета, передающего этап согласования (negotiate_step)
noPeer	23	Нет партнёра для согласования (req_negotiate, negotiate_step)
CBORfail	24	Отказ при декодировании CBOR (req_negotiate, negotiate_step, synchronize)
invalidNeg	25	Непригодное сообщение Negotiate (req_negotiate, negotiate_step)
invalidEnd	26	Непригодное сообщение о завершении (req_negotiate, negotiate_step)
noNegReply	27	Нет отклика на этап согласования (req_negotiate, negotiate_step)
noValidStep	28	Нет пригодного отклика на этап согласования (req_negotiate, negotiate_step)
sockErrWait	29	Ошибка сокета, передающего сообщение об ожидании (negotiate_wait)
sockErrEnd	30	Ошибка сокета, передающего сообщение о завершении (end_negotiate, send_invalid)
IDclash	31	Конфликт Session ID во входящем запросе (listen_negotiate)
notSynch	32	Нет задачи для синхронизации (synchronize, get_flood)
notFloodDisc	33	Нет лавинной рассылки и отклика на обнаружение (synchronize)
sockErrSynRq	34	Ошибка сокета, передающего запрос синхронизации (synchronize)
noListener	35	Нет прослушивающей стороны для синхронизации (synchronize)
noSynchReply	36	Нет ответа на запрос синхронизации (synchronize)
noValidSynch	37	Нет пригодного ответа на запрос синхронизации (synchronize)
invalidLoc	38	Недействительный локатор (flood)

Благодарности

Существенные предложения внесли Ignas Bagdonas, Carsten Bormann, Laurent Ciavaglia, Roman Danyliw, Toerless Eckert, Benjamin Kaduk, Erik Kline, Murray Kucherawy, Paul Kyzivat, Guangpeng Li, Michael Richardson, Joseph Salowey, Éric Vyncke, Magnus Westerlund, Rob Wilton и другие члены рабочей группы IESG ANIMA.

Адреса авторов

Brian E. Carpenter
School of Computer Science
University of Auckland
PB 92019
Auckland 1142
New Zealand
Email: brian.e.carpenter@gmail.com

Bing Liu (editor)
Huawei Technologies
Q14, Huawei Campus
No.156 Beiqing Road
Hai-Dian District, Beijing
100095
China
Email: leo.liubing@huawei.com

Wendong Wang
BUPT University
Beijing University of Posts & Telecom.
No.10 Xitucheng Road
Hai-Dian District, Beijing 100876
China
Email: wdwang@bupt.edu.cn

Xiangyang Gong
BUPT University
Beijing University of Posts & Telecom.
No.10 Xitucheng Road
Hai-Dian District, Beijing 100876
China
Email: xygong@bupt.edu.cn

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru