Instituto Tecnológico de Buenos Aires
Protocolos de Comunicación
Grupo 7

Serpe, Octavio Javier (60076)
Rodríguez, Manuel Joaquín (60258)
Arca, Gonzalo (60303)
Parma, Manuel Félix (60425)
Junio 2021

Proxy Configuration and Monitoring Protocol -- PCAMP 1.0

Estado de este memorándum

Este documento especifica un protocolo realizado para el proyecto de Trabajo Práctico Especial de la materia Protocolos de Comunicación en el Instituto Tecnológico de Buenos Aires.

1. Introducción

1.1. Propósito

El Protocolo de Configuración y Monitoreo de Proxy (PCAMP, por sus siglas en inglés Proxy Configuration and Monitoring Protocol) es un protocolo de nivel de aplicación diseñado para trabajar en conjunto con un proxy HTTP y obtener métricas y realizar cambios de configuraciones pertenecientes al mismo.

1.2. Requisitos

Las palabras clave "DEBE", "NO DEBE", "OBLIGATORIO", "DEBERÁ", "NO DEBERÁ", "DEBERÍA", "NO DEBERÍA", "RECOMENDADO", "PUEDE" y "OPCIONAL" en este documento serán interpretadas como se describe en el RFC 2119 [1].

1.3. Terminología

Esta especificación utiliza una serie de términos para referir a los roles asumidos por los participantes y los objetos de la comunicación a través de PCAMP.

mensaje
La unidad básica de la comunicación en PCAMP, consistiendo en una secuencia de campos definidos en la sección 3.

petición
Un mensaje de petición PCAMP, definido en la sección 4.

respuesta
Un mensaje de respuesta PCAMP, definido en la sección 5.

cliente PCAMP
Un programa que se encarga de enviar peticiones de métricas o de cambios de configuraciones en un proxy HTTP.

servidor PCAMP
Un programa de aplicación que recibe peticiones de clientes PCAMP y responde según el contenido de las peticiones.

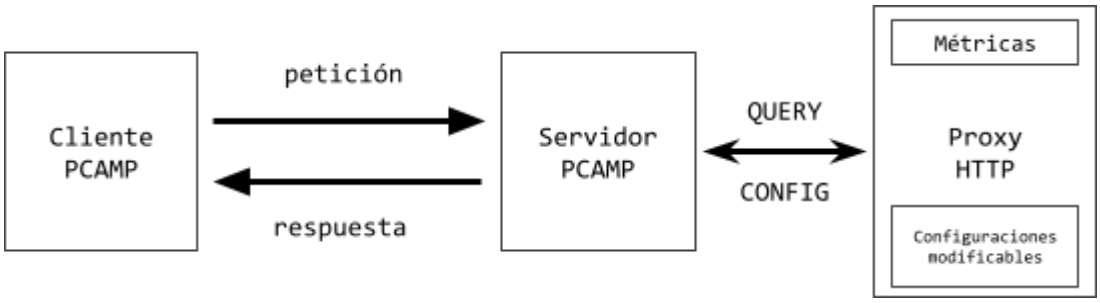
proxy HTTP
Se toma la definición de proxy de la terminología del RFC 2616. [2]

1.4. Operación general

El protocolo PCAMP se basa en la abstracción de petición y respuesta. Un cliente envía una petición utilizando la operación REQUEST y especificando un método, y luego el servidor responderá utilizando la operación RESPONSE y un STATUS CODE indicando si la operación fue exitosa o errónea.

La comunicación se da a través del protocolo de transporte UDP [3] por defecto en el puerto 9090, pero otros puertos PUEDEN ser utilizados. DEBE utilizarse el orden big-endian a la hora de enviar y recibir los mensajes.

El principal esquema de uso del protocolo DEBERÍA ser el siguiente:



2. Parámetros del protocolo

A continuación se enumeran los parámetros que varían en cada petición y respuesta en el uso de PCAMP 1.0

2.1. Operaciones

REQUEST
: La operación de un mensaje PCAMP se denomina REQUEST cuando la misma es un mensaje de petición que se dirige de un cliente a un servidor PCAMP. Su contenido se detalla en la sección 4.

RESPONSE
: La operación de un mensaje PCAMP se denomina RESPONSE cuando la misma es un mensaje de respuesta que se dirige de un servidor a un cliente PCAMP. Su contenido se detalla en la sección 5.

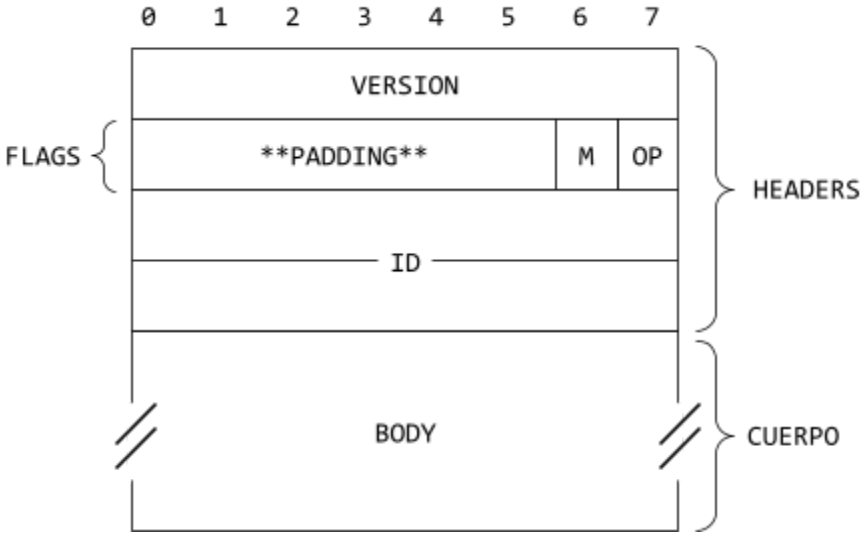
2.2. Métodos

QUERY
: El método QUERY es utilizado para obtener métricas registradas en el proxy HTTP. DEBE solicitar 1 dato a la vez. Su formato se especifica en la sección 4, y los respectivos datos soportados en la sección 6.

CONFIG
: El método CONFIG es utilizado para modificar un conjunto de configuraciones de un proxy HTTP. DEBE solicitar la modificación de 1 parámetro a la vez. Su formato se especifica en la sección 4 y los respectivos parámetros soportados en la sección 7.

3. Mensaje PCAMP

Cada mensaje PCAMP consiste en una sección de headers y un cuerpo de la siguiente forma:



La sección de headers tiene un ancho fijo de 4 octetos, cuyos valores se especifican en la sección 3.1. El cuerpo tiene un ancho variable dependiendo del método y de la operación.

3.1. Headers de un paquete PCAMP

El formato de los headers es el siguiente:

VERSION 1 octeto utilizado para especificar la versión de PCAMP utilizada por el cliente o servidor que envía.

FLAGS 1 octeto conteniendo el método (M) y la operación (OP) cada uno en un bit en la parte menos significativa del octeto.

Tanto servidor como cliente DEBEN ignorar lecturas de los bits a la izquierda de dichos campos (**PADDING** en la figura).

M 1 bit utilizado para representar el método.

0 representa QUERY.

1 representa CONFIG.

El servidor DEBE incluir en la respuesta el mismo valor de M que vino en la petición del cliente.

OP 1 bit utilizado para representar la operación.

0 representa REQUEST.

1 representa RESPONSE.

El cliente DEBE asignar 0 a este valor y el servidor DEBE asignar 1.

ID 2 octetos utilizados para identificar el intercambio REQUEST-RESPONSE entre cliente y servidor.

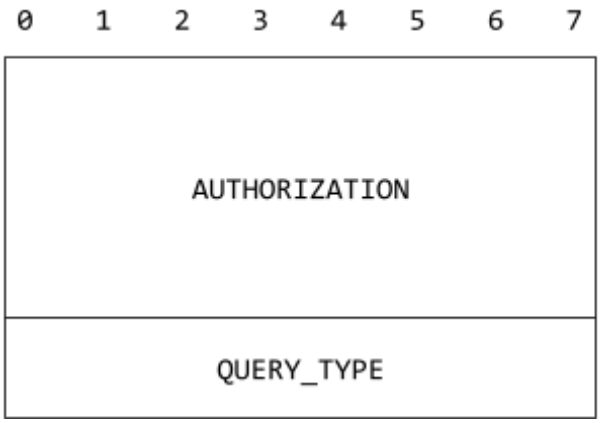
Cada respuesta DEBE incluir el ID que vino en la petición. El cliente PUEDE elegir el criterio que desee para la elección de dicho valor.

3.2. Cuerpo de un paquete PCAMP

El cuerpo de un paquete PCAMP varía por método y por operación. Sin embargo, todos los paquetes de operación REQUEST DEBEN contener un campo de ancho fijo AUTHORIZATION, el cual se detalla en la sección 9. Asimismo todos los paquetes de operación RESPONSE DEBEN contener un campo de STATUS_CODE, cuyos valores se especifican en detalle en la sección 8.

3.2.1. Cuerpo de un paquete PCAMP con método QUERY

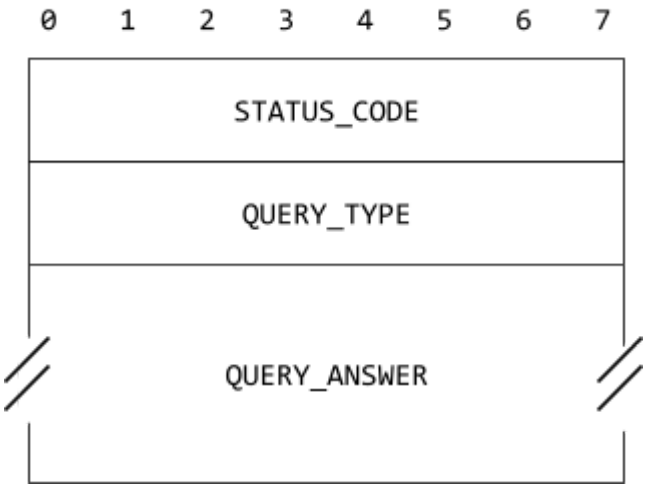
Para un paquete con método QUERY y operación REQUEST, el cuerpo tiene la siguiente forma:



AUTHORIZATION 32 octetos conteniendo una contraseña proporcionada por el cliente, la cual DEBE estar hasheada mediante el algoritmo de hashing seguro SHA-256.

QUERY_TYPE 1 octeto utilizado para aclarar el tipo de métrica que desea obtener del proxy HTTP. Sus tipos se aclaran en la sección 6.

Para un paquete con método QUERY y operación RESPONSE, el cuerpo tiene la siguiente forma:



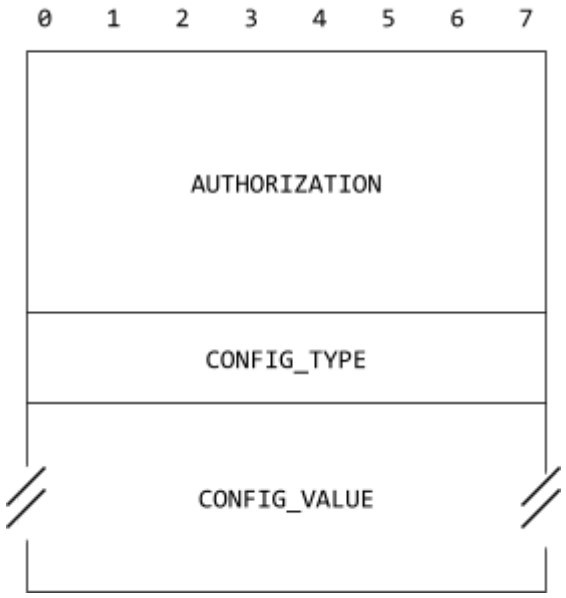
STATUS_CODE 1 octeto utilizado para indicar el estado de la operación solicitada al servidor.

QUERY_TYPE 1 octeto conteniendo el tipo de métrica contenida en el campo QUERY_ANSWER. En caso de ser un tipo válido, el mismo DEBE incluir el mismo QUERY_TYPE que vino en la petición.

QUERY_ANSWER Campo de ancho variable que contiene la métrica solicitada en la petición.

3.2.2. Cuerpo de un paquete PCAMP con método CONFIG

Para un paquete con método CONFIG y operación REQUEST, el cuerpo tiene el siguiente formato:

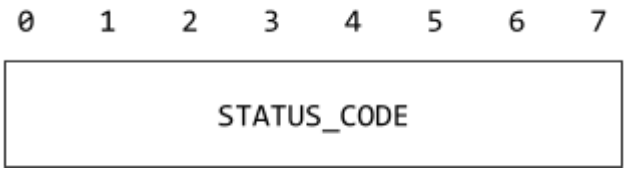


AUTHORIZATION 32 octetos conteniendo una contraseña proporcionada por el cliente, la cual DEBE estar hasheada mediante el algoritmo de hashing seguro SHA-256.

CONFIG_TYPE 1 octeto utilizado para indicar el tipo de configuración que se desea modificar en el proxy HTTP.

CONFIG_VALUE Campo de ancho variable que contiene el nuevo valor de la configuración a modificar en el proxy HTTP.

Para un paquete con método CONFIG y operación RESPONSE, el cuerpo tiene la siguiente forma:



STATUS_CODE 1 octeto utilizado para indicar el estado de la operación solicitada al servidor.

4. REQUEST

4.1. Política de mejor esfuerzo desde el cliente

El cliente DEBERÍA estar preparado contra la posible pérdida de paquetes dado que el protocolo permite la adopción de una política de mejor esfuerzo en lo que respecta a la recepción de peticiones y envío de respuestas en el servidor. Es RECOMENDADO utilizar herramientas como timeouts en conjunto con retransmisiones para enfrentar tal situación.

4.2. Integridad de campos en REQUEST

A la hora de realizar una petición a un servidor de PCAMP, un cliente DEBE utilizar los valores de QUERY_TYPE y CONFIG_TYPE especificados en la secciones 6 y 7, respectivamente. Al mismo tiempo, los tamaños del campo CONFIG_VALUE DEBEN respetar los valores aclarados en la sección 7.2.

Por otro lado, el cliente DEBE autenticarse en cada petición que envíe al servidor. Más sobre el método de autenticación en la sección 9.

5. RESPONSE

5.1. Política de mejor esfuerzo desde el servidor

En cuanto a la recepción de una petición y al envío de una respuesta desde el punto de vista del servidor PCAMP, éste PUEDE tomar una política de mejor esfuerzo. Esto implica que si llegan a haber errores internos al intentar realizar dichas acciones, el servidor PUEDE no reintentar la operación y/o garantizar que el paquete se envíe de cualquier otra manera posible.

5.2. Integridad de QUERY_ANSWER

El servidor DEBE respetar los tamaños indicados en las sección 6. a la hora de asignar el valor de QUERY_ANSWER en el paquete de respuesta.

5.3. Manejo de peticiones con parámetros válidos

Cuando se trata de peticiones cuyos parámetros son válidos tanto en los headers como en el cuerpo, el servidor debe realizar la operación especificada por el método, y asignar en STATUS_CODE el valor 0 (SUCCESS). Para ver todos los códigos de estado, consultar la sección 8.

5.4. Manejo de peticiones con parámetros inválidos desde el servidor

Para cada una de las siguientes situaciones, el servidor DEBE utilizar los códigos mencionados en la sección 8, y PUEDE enviar un cuerpo únicamente con STATUS_CODE y ningún otro campo más, independientemente del método especificado.

5.4.1. Validación de versión de PCAMP

Si la versión contenida en el paquete PCAMP es distinta a la versión soportada por el servidor, éste DEBE asignar el código 2 en el campo STATUS_CODE (UNSUPPORTED_VERSION en sección 8).

5.4.2. Validación de operación

Si en la petición llega el campo OP asignado con 1, es decir si llega una RESPONSE, el servidor DEBE responder con los mismos headers de la petición y el STATUS_CODE con el valor en 6 (BAD_REQUEST en sección 8).

5.4.3. Validación de QUERY_TYPE y CONFIG_TYPE

En cuanto al envío de respuestas a peticiones, el servidor DEBE comparar los valores de QUERY_TYPE y CONFIG_TYPE contra aquellos mencionados en las secciones 6 y 7.

En el caso de que el valor QUERY_TYPE no coincida con ningún otro de la sección 6, el servidor DEBE asignar en STATUS_CODE el valor 3 (UNSUPPORTED_QUERY_TYPE en sección 8). Análogamente, si el valor de CONFIG_TYPE tampoco tiene coincidencias en la sección 7, el servidor DEBE asignar en STATUS_CODE el valor 4 (UNSUPPORTED_CONFIG_TYPE).

5.4.4. Validación de CONFIG_VALUE

Si algún valor dentro del campo CONFIG_VALUE no respeta las restricciones para cada CONFIG_TYPE especificadas en la sección 7, el servidor DEBE responder con STATUS_CODE asignado en 5 (INVALID_CONFIG_VALUE en la sección 8).

5.4.5. Validación de AUTHORIZATION

Si el valor del campo AUTHORIZATION no cumple las condiciones requeridas en la sección 9, el servidor DEBE responder con STATUS_CODE asignado en 1 (AUTH_ERROR en sección 8).

5.5. Manejo de respuestas con parámetros inválidos desde el cliente

En caso de que el cliente llegue a recibir una respuesta con parámetros inválidos, como un ID distinto al de la petición enviada, una versión de PCAMP incompatible, una REQUEST, o un

método distinto al solicitado, el cliente DEBE ignorar el paquete.

5.6. Validación de QUERY_ANSWER desde el cliente

El cliente DEBE respetar los tamaños de QUERY_ANSWER especificados en la sección 6. a la hora de leer tal campo de una respuesta a una QUERY. Dada esta condición, el cliente PUEDE no validar la respuesta que llegue en el campo QUERY_ANSWER.

6. Restricciones sobre QUERY_TYPE y QUERY_ANSWER

A continuación se exponen los posibles valores de QUERY_TYPE, cada uno con su respectivo tamaño de QUERY_ANSWER.

Valor de QUERY_TYPE	Métrica correspondiente	Tamaño en octetos (ancho fijo)
0	Cantidad de conexiones históricas en proxy HTTP	8
1	Cantidad de conexiones concurrentes en proxy HTTP	8
2	Cantidad de octetos transferidos de proxy a clientes y de proxy a servidores	8
3	Cantidad de octetos transferidos de proxy a servidores	8
4	Cantidad de octetos transferidos de proxy a clientes	8
5	Cantidad de octetos transferidos a través del método de HTTP CONNECT [4]	8

7. Restricciones sobre CONFIG_TYPE y CONFIG_VALUE

Valor de CONFIG_TYPE	Significado	Valores permitidos	Tamaño en octetos (ancho fijo)
0	Tamaño de buffer de entrada/salida del proxy HTTP	1 - 65535	2
1	Cantidad máxima de clientes conectados concurrentemente permitida	0 - 509	2
2	Activar/desactivar captura de credenciales de proxy	0 (apagado) 1 (encendido)	1
3	Dirección IP de servidor DoH	Ver sección 7.1.	17
4	Puerto de servidor DoH	0 - 65535	2
5	Nombre de host de servidor DoH	Ver sección 7.2.	256

7.1. Formato de CONFIG_VALUE para dirección IP de servidor DoH

El cliente DEBE utilizar un ancho fijo de 17 octetos al enviar una nueva dirección para el servidor DoH utilizado por el proxy. En el primer octeto, debe insertar 0 si se trata de una dirección IPv4, y 1 en el caso de una dirección IPv6. A partir del siguiente octeto debe volcar la dirección que corresponda.

El servidor DEBE ignorar los octetos restantes del campo si recibe una dirección IPv4.

7.2. Formato de CONFIG_VALUE para nombre de host de servidor DoH

El cliente DEBE utilizar un ancho fijo de 256 octetos al enviar un nuevo nombre de host para el servidor DoH utilizado por el proxy. Dentro del ancho fijo debe incluir la cadena de caracteres a partir del primer octeto. La cadena DEBE delimitarse con el octeto nulo al finalizar, lo cual obliga al cliente a utilizar una cadena de máximo 255 octetos, sin incluir al nulo.

8. STATUS-CODE

Código de estado	Nombre de estado	Descripción de estado
0	SUCCESS	La petición se pudo resolver exitosamente
1	AUTH_ERROR	La contraseña no coincide con la del servidor
2	UNSUPPORTED_VERSION	La versión de PCAMP no coincide con la del servidor
3	UNSUPPORTED_QUERY_TYPE	El QUERY_TYPE provisto no es soportado por el servidor
4	UNSUPPORTED_CONFIG_TYPE	El CONFIG_TYPE provisto no es soportado por el servidor
5	INVALID_CONFIG_VALUE	El CONFIG_VALUE provisto no es válido según la sección 7
6	BAD_REQUEST	Cualquier caso de errores de peticiones que no estén abarcados en los anteriores estados
7	INTERNAL_SERVER_ERROR	Error en el servidor al querer resolver la petición

9. AUTHORIZATION

Un servidor PCAMP DEBE poseer un sistema de autenticación. Éste DEBE basarse en el intercambio de una contraseña hasheada a través del algoritmo de hashing seguro SHA-256 cada vez que un cliente realiza una petición. El servidor luego debe tomar su contraseña, aplicarle el mismo algoritmo de hashing, y comparar la cadena generada con la cadena de la petición. Si no coinciden, el servidor DEBE responder con STATUS_CODE 1 (AUTH_ERROR en sección 8).

10. Referencias

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, Marzo 1997.

[2] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T. "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, Junio 1999.

[3] Postel, J., "User Datagram Protocol", RFC 768, Agosto 1980.

[4] Fielding, R., Reschke, J., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, Junio 2014.