Edition 2024

SL DPI

Extension Dictionary

slinkin.tech

Contents

Extension Mechanism

Extension mechanism is an important technical advantage of **DC Engine** framework. That feature provides the opportunity for users/developers extend **DC Engine** functionality without modifying the core code base. Extension is a code snippet which takes Packet, Flow context and Global entities and does code execution. In the most cases, it addes new fields to Global object and they can be used in the future for classification, logging and any other purposes.

The extension mechanism was implemented for:

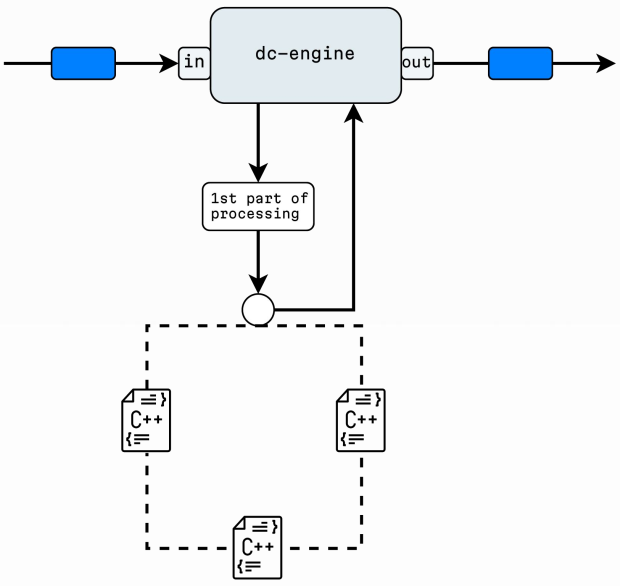
- Implement logic which is not fully related to protocol dissection, flow updating, session control, etc.
- Allow user/developer to extend the base **DC Engine** functionality on their own

Types:

- **In-built** extensions
- **User** extensions

To demonstrate what extension can do, let's consider **DNS Response Name in-buit** extension. **DNS** protocol uses compression of response names. The one part of the name can be presented as an **ASCII** string, another part can be presented as an offset to another **ASCII** string, etc. It means, after protocol dissection a user/developer cannot easily get all response names, because they are allocated as not a continuous char buffer.

DNS Response Name extension analyzes **DNS** layers and decompresses (concat labels which are located in different places of protocol layer) response names, to provide the opportunity to easily use values for classification, logging, and other purposes.



Users/developers are not limited by **in-built** extensions and can add own code snippets using **DC Engine API**.

In-built extensions

> HTTP METADATA

The extension analyzes **HTTP** message(s) to define Request/Response property and detect Pipeline messages.

Name	Type	Length	Mask	Description
is_pipeline	**uint8**	**1**	ffffffffffffff	Multiple http requests to be sent over a single tcp connection.
is_request	**uint8**	**1**	ffffffffffffff	Is set when http meessage is request.

> HTTP HOST

The extension implements simple logic – it searches Host header and stores value of that. It is implemented because **HTTP** usually contains more than one header and when http.header is used in classification rule it leads to checking **ALL** headers. Since **HTTP** Host header value is the most usable, much better to store it in a different place and don't search it everytime among the headers.

Name	Type	Length	Mask	Description
host	**ascii-string**	**0**	ffffffffffffff	Value of HTTP Host header.

> DNS RESPONSE NAME

The extension decompress (concat labels which are located in different places of protocol layer) response names of **DNS** message.

Name	Type	Length	Mask	Description
response_name	**ascii-string**	**0**	ffffffffffffff	Processed dns response name – concatenated labels, resolving offsets, etc.

> TLS CERTIFICATE

At this moment, **TLS Certificate** dissection is not supported, but **common_name** and **general_dns_name** fields play significant role in classification. The extension extracts that fields from **TLS Certificate**(s).

Name	Type	Length	Mask	Description
general_dns_name	**ascii-string**	**0**	ffffffffffffff	The dNSName certificate field. Belongs to extension section.
common_name	**ascii-string**	**0**	ffffffffffffff	The CommonName certificate field.

> DNS CACHE

DNS Cache is one of the most usable classification method. The extension stores dns response names and related IP addresses to table and allow user/ developer to check IP address of new flowaccording to the table. It also controls TTL of each response name to remove expired records.

Name	Type	Length	Mask	Description
ipv4_response _name	**ascii-string**	**0**	ffffffffffffff	Cached dns response name for destination ipv4 address.

> DNS RESPONSE NAME

Quic Cache extension implements logic of storing Connection ID(s) of the same quic session. Unfortunatelly, New Connection Id frames are encrypted after QUIC Initial(s) and to use this extension, a user/developer has to decrypt QUIC payload.

Name	Type	Length	Mask	Description
session_flow_ tag	**uint64**	**8**	ffffffffffffff	Flow tags of the same quic session.

slinkin.tech