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:

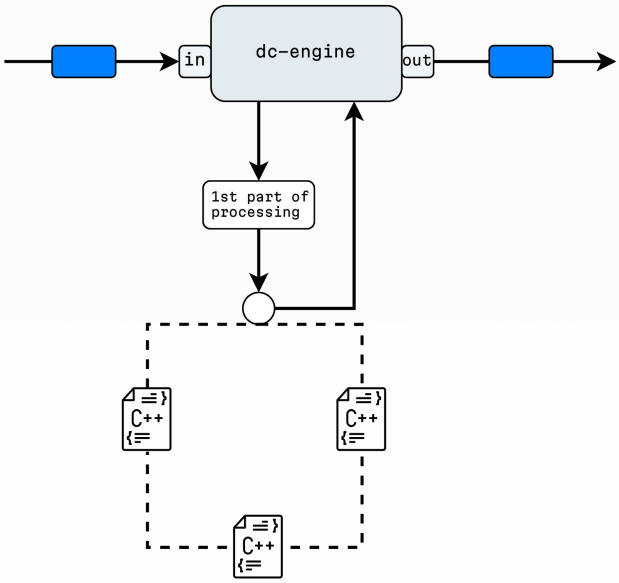
- Implement logic which is not fully related to protocol dissection, flow updating, session control, etc.
- Allow user/developer to extend the base **DC Engine** functionality on their own

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	Multiple	Description
is_pipeline	**uint8**	**1**	ffffffffffffffff	false	Multiple http requests to be sent over a single tcp connection.
is_request	**uint8**	**1**	ffffffffffffffff	false	Is set when http meessage is request.

> HOST NAME

The extension implements simple logic - it extracts the values from protocol fields that store host names, e.g., the value of the HTTP1/.

* Host header or server_name value of TLS/ DTLS protocols, etc.

Name	Type	Length	Mask	Multiple	Description
host	**ascii-string**	**0**	fffffffffffffff	true	Extracted field value from TLS/ DTLS SNI, HTTP/SSDP Host header, TLS Certificate Common Name, HTTP/2 authority header value.

> 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	Multiple	Description
response_name	**ascii-string**	**0**	fffffffffffffff	true	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	Multiple	Description
general_dns_name	**ascii-string**	**0**	fffffffffffffff	true	The dNSName certificate field. Belongs to extension section.
common_name	**ascii-string**	**0**	fffffffffffffff	true	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	Multiple	Description
ipv4_response_name	**ascii-string**	**0**	ffffffffffffffff	true	Cached dns response name for destination ipv4 address.

> PROTOCOL TREE

Protocol Tree is a simple extension that goes through all layers and concatenates protocol names, e.g., ethernet.ipv4.tcp.http.

The proto_tree is not multiple field - if the packet has more than one frame, the protocol names are concatenated within the single string.

Name	Type	Length	Mask	Multiple	Description
proto_tree	**ascii-string**	**0**	ffffffffffffffff	false	The protocol tree string.

> DATA STRUCTURE

Data Structure extension implements the logic of checking the payload data structure, e.g. when some first 1/2/4 bytes have a value that specifies the length of the payload. It might be useful to classify services that do not use public protocol.

Name	Type	Length	Mask	Multiple	Description
first_byte_payload_length	**uint64**	**8**	ffffffffffffffff	false	The flag field. It is set when the first byte in the payload data specifies the length of the rest payload data.
first_2byte_payload_length	**uint64**	**8**	ffffffffffffffff	false	The flag field. It is set when the first 2 bytes in the payload data specifies the length of the rest payload data. (Little-Endian order)

> HTTP/2 FIELD BLOCK

HTTP/2 Field Block extension decodes (HPACK) HTTP/2 field blocks and extract HTTP/2 headers.

Name	Type	Length	Mask	Multiple	Description
header_field	**ascii-string**	**0**	ffffffffffffff	true	The http/2 protocol header block field: name:value.
push_promise_field	**ascii-string**	**0**	ffffffffffffff	true	The http/2 protocol push-promise block field: name:value.
continuation_field	**ascii-string**	**0**	ffffffffffffff	true	The http/2 protocol continuation block field: name:value.

slinkin.tech