Packet Payload Processing (PPP)

|  |  |
| --- | --- |
| Version | 0.1 |
| Author | Tom Sumardi |
| TODO |  |

Table of Contents

Introduction 1

Abbreviations 1

Conventions/Requirements 1

Summary 1

Components 2

Architecture 2

Component descriptions: 2

# Introduction

This documentation intended to serve the purpose of giving detail level architecture description.

# Abbreviations

TBD

# Conventions/Requirements

* Performance bound
* Bidirectional communication interface from MS to PPP will be through AMQP
* Unidirectional communication interface from PPP to PHS will be through ZeroMQ
* PHS is a service that plugs into PPP component
* Packet outgoing interface will go through different interface than packet incoming interface
* Default PPP plugin will be HTTP processing

# Summary

PPP operation only involves user space and acts as plugin to the IPPS with “HTTP processing” as the default plugin.

PPP can be null operation if no payload parsing to be done on the packet and by default set to process HTTP packet. It parses, dissects and applies regex on the packet payload (L4 payload) at user space level. It performs regex matching based on configuration given and determines if the packet needs to be passed to the PHS or not.

Example of PPP acting as packet injection plugin operation:

* above L4 decoding
* above L4 packet filtering
* session correlation

# Components

null

IPPS Thread

HTTP

plugin

Other…

ZeroMQ

PHS

Logging

Config/Register

AMQP

Figure 1 (component stack)

* Background configuration/registration thread
* L4 Packet Processing Plugin
* ZeroMQ communication
* Logging

# Architecture

## Component descriptions:

1. Background configuration/registration thread

Background thread is responsible for configuring and registering PPP component, which

uses AMQP as communication channel to the MS.

Responsibilities:

* Registers plugin information
* Configure/load plugin (default: HTTP plugin)

This thread will be idle once configuration and registration have been successfully applied. Configuration can only be done once PPP component has been successfully registered.

state machine:

|----------------| |-----------|

down -> up -> registering -> registered -> loading -> loaded

| |

error

Below is the bi-directional communication between MS and PPP:

PPP -> MS registration:

* PPP uuid
* status (state-machine): registering, registered, error

MS -> PPP registration:

* status (state-machine): registering, registered, error

MS -> PPP configuration:

* plugin:
  + L4-processor/filter binary
  + Name
  + ZeroMQ metadata
  + Version
  + PHS uuid (PHS channel)
  + status (state-machine): loading, loaded, error

PPP -> MS configuration:

* status (state-machine): loading, loaded, error

1. L4 Packet Processing Plugin

The plugin will perform L4 packet payload processing, which can be swapped based on request from the MS. Minimally, the plugin will be a binary L4-processor/filter coming from MS as input, which perform packet processing with processing logic to decide whether packets need to be dropped or passed to PHS, which will be the next and last processing in packet life cycle.

IPPPS -> PPP packet:

- pkt payload

1. ZeroMQ communication

The uni-direction communication channel between PPP and PHS will be done through zeroMQ. The relationships between PPP with PHS service will be one-to-one relationships with one IPPS thread owns one zeroMQ communication channel. The channel relationships will be based on zeroMQ metadata being supplied from MS.

1. Logging

TBD