Table des matières

[1 Objectif 1](#_Toc463700662)

[2 Reception des paquets 3](#_Toc463700663)

[2.1 Protocoles pris en charge 3](#_Toc463700664)

[2.2 Laxisme en reception 3](#_Toc463700665)

[3 Contraintes de développements 4](#_Toc463700666)

[3.1 Licence 4](#_Toc463700667)

[3.2 Architecture et OS 4](#_Toc463700668)

[3.3 Intégration à Wireshark 4](#_Toc463700669)

[3.4 Pratique d’utilisation 5](#_Toc463700670)

[4 Définition du protocole propriétaire 6](#_Toc463700671)

[4.1 Analyse protocolaire 6](#_Toc463700672)

[4.2 Fichier de configuration 6](#_Toc463700673)

[5 Livrables 7](#_Toc463700674)

[5.1 Software 7](#_Toc463700675)

[5.2 Documentations 7](#_Toc463700676)

# Objectif

L’objectif est d’enrichir l’outil d’analyse de protocole Wireshark avec un nouveau plugin qui prend en charge un protocole de communication propriétaire.

# Reception des paquets

## Protocoles pris en charge

Le plugin se devra de filtrer pour analyse, les trames des protocoles suivants:

* TCP
* UDP
* AFDX
* Etherfly

## Laxisme en reception

Il n’y a pas de qualité de services (QoS). Les QoS sont gérées par ailleurs par le hardware.

Le plugin Wireshark sera utilisé comme:

* outil de post-analyse de fichiers .pcap
* outil de live-monitoring pour une vision global des échanges sur le réseau.

En live-monitoring, il est toléré que le plugin Wireshark perde quelques trames pendant son analyse.

# Contraintes de développements

## Licence

Lié un module ou plugin statiquement ou dynamiquement avec une ABI de Wireshark est considéré comme un travail combinatoire basé sur le logiciel Wireshark. Ce dernier étant sous licence GPL, le plugin doit l’être aussi.

Le plugin sera donc sous license GPL.

## Architecture et OS

Le plugin sera lancé sur une plateforme de type PC avec architecture 32 et 64 bits.

Il sera supporté par les système d’exploitation suivants:

* GNU/Linux 32 et 64 bits
* Cygwin pour les versions Windows 32 et 64 bits

## Intégration à Wireshark

Maintenable:

Pour des questions de maintenabilité, le plugin s’intégrera simplement à Wireshark.  
L’intégration comprenant l’étape de compilation et de chargement du plugin.  
Eviter au maximum de recompiler les sources du plugin avec les sources complètes de Wireshark.

Solutions envisagées:

* Soit une bibliothèque partagée (.SO)
* Soit en script LUA

Dans le cas de la bibliothèque partagée:

* les sources seront en langage C
* la complation requiera certains fichiers header de Wireshark et non les sources complètes
* la compilation s’execu-tera avec l’outil CMake

## Pratique d’utilisation

Le plugin sera utilisé sur une version graphique de Wireshark pour bénéficier de son interface utilisateur, d’options de filtrage dynamique et de visualisation de capture.

# Définition du protocole propriétaire

## Analyse protocolaire

Travaillant sur des protocoles connus de Wireshark (Excepté AFDX), leurs paquets sont donc déjà disséqués et pris en charge afin d’être validé. Nous nous placerons alors au dessus de la couche de transport pour les protocoles TCP/UDP et au dessus de la couche internet (IP) pour AFDX/Etherfly, afin d’y effectuer l’analyse du protocoles propriétaires à travers leurs payloads.

## Fichier de configuration

La définition d’un protocole propriétaire se fera à travers un fichier de configuration (un pour chaque protocole).

Le format du fichier de configuration devra être simple à mettre en œuvre par un développeur tout en étant complet.  
Ajouter à cela qu’il doit être peu verbeux et facile d’appréhension.

La configuration permettra :

* Une écoute sur un port précis ou sur une plage de ports;
* Un découpage flexible des différents champs d’un payload de taille variable.

# Livrables

## Software

Le Plugin Wireshark fonctionnel sera livré avec ses sources et son code commenté.

## Documentations

* Document de conception de réalisation du plugin
* Document d’installation (utilisateur):
  + Installation et configuration du plugin suivant les OS
  + La compatibilité avec les versions de WireShark
* Document de mise en oeuvre (développeur), orienté maintenance/évolutions:
  + Installation du compilateur (limité aux modules nécessaires),
  + Comment compiler et déployer le plugin ?
  + Comment modifier le plugin en cas d’évolution du protocole ?