Techniques of the Communication Data Processing

The process of analyzing a new communication protocols is a time intensive task that includes studying data formats, developing prototype parser, testing prototypes and repeating the cycle until a robust parser is full developed. Domain specific protocol parsing tools help in the process by simplifying tasks with parsing and analyzing models.

For software developers that work on communication data processing, dealing with raw bits and bytes in data streams can be unnecessary task every time a new parser development project shows up. Even though developing against bits and bytes gives better control, a higher level abstraction of data stream representation such as [1],[2]and [3]can help avoid complexities and implementation bugs. Protocol handlers such as Googles Protocol Bufferset al [4] enable direct (de)serialization of memory objects to and from network streams. Other such serialization implementations in major computer languages can also be used in this regard. But, since these serializes have strict binary data representation that is optimized for smaller disk space and lower bandwidth usage, developing parsers with them is difficult or impossible.

Protocol definition languages like Generic Application Level Protocol Analyzer (GAPA)et al[5] are better suited for the task of parser development because they enable to describe a protocol with pseudo codes or custom syntaxes which later can be exported to a language of interest such as C++ or compile with their own compilers and be deployed directly to the target systems. Each packet filter discussed in [3] is represented by a context-free grammar, whose language is the set of packets a filter should accept so that packet filters can be formulated through a general, well defined specification. Such languages are domain specific and they are intended to simplify creating new or existing communication protocols. But they can not be directly used if we want to analyze, visualize, route and retransmit information in a network. Bro et al [6] is more related to our model in that it tries to create a complete solution of parsing and monitoring of a network stream. But, Bro is inclined towards TCP/IP protocols and its intention is for detecting and reporting irregularities, which is only a part of the many goals of the model proposed.

The generic protocol representation and data routing model described here tries to create an abstraction layer on top of raw data streams. Since it supports most C like data types, it can act as an interpreted protocol language for defining new protocol formats or for parsing and analyzing most of existing protocol definitions. It doesn’t create additional memory data structure, rather the model introduces entities which are virtual objects that have their value evaluated at runtime, when a user or developer specifically requests for it. This is good design choice when it comes to conserving memory; a precious resource in communication analysis field.

This new model can be used to create intelligence gathering tools by specifying policies as data parsing formats, as data storage mechanism by defining specifics to be included out of a given data stream, as an information relaying platform.