I. INTRODUCTION AND MOTIVATION

Cisco® NetFlow protocol\(^1\) is a valid alternative to Deep Packet Inspection (DPI) for network monitoring, since it provides a lightweight picture of exchanged traffic, avoiding the burden of payload access, like privacy concerns and high resource demand. The NetFlow protocol is able to condensate in a single record (called netflow) a unidirectional sequence of packets that share the source-destination addresses and that have the same ports, IP protocol, ingress interface, and IP type of service. Moreover, several other valuable data are included in a flow, like timestamp, duration, number of exchanged packets, and number of transmitted bytes. Actually, NetFlow can be used for many applications, including Intrusion Detection Systems [1], DoS [2] and anomaly detection [3] and more.

DPI and netflow analysers are generally available as closed-source software, since part of commercial network monitoring packages. This has lead academic researchers to perform netflows analysis with ad-hoc, home-made software. In fact, nfdump, that is the most spread open-source project for netflow analysis, has several aspects that limit its adoption. For example, nfdump has a modular structure, but each plugin has to be executed independently on each netflow block (execution time linear with the number of plugins). Moreover, it does not properly manage netflow timeouts and does not allow the addition of a new plugin at run time. Those limitations make it unsuitable for time based plugins and for large scale network analysis (i.e. Autonomous System). This is why we propose FAN, an open-source, general-purpose and lightweight framework for fast netflows analysis. FAN is written in C and can run any kind of plugins for slotted netflow analysis. For example, we already developed some plugins for anomaly detection (Section III). It has a plugin manager able to customize the plugin dependencies, in order to optimise the computations during the analysis. Moreover, it pays a particular attention to netflow timeout management, that is a critical aspect of the Netflow technology.

II. FAN: DESIGN AND PRINCIPLES

The analyser is composed by three elements: the collector, the plugin manager and the logging system. The collector receives raw netflows and packs them in blocks according to timeouts. Then, each block of netflows is passed to the plugin manager, that is in charge to optimise the operation sequences for netflows analysis. Both collector and plugin manager reports to the logging system that will present the analysis results of each plugin. In the following we describe in greater detail the two more interesting elements of our framework, the timeout and the plugin managers.

A. Timeout management

A new flow is generated by monitoring probes under two different conditions: when the peers explicitly terminate the connection according to the protocol, and when a timeout expires. Netflow probes use two independent timeouts. The inactive timeout (\(\theta_i\)) triggers when the peers do not exchange packets for \(\theta_i\) seconds and the connection has not been terminated. For each newly exchanged packet, the probe resets \(\theta_i\). The active timeout is used to break up long-lived flows into several fragments, and triggers each \(\theta_a\) seconds until two peers exchange packets at a high rate on the same flow. Tuning \(\theta_a\) parameter can be tricky for network administrators. In fact, long-lived flow fragments can alter the real representation of the data crossing the network. Usually, low active timeout values are used for detection purposes (i.e. high responsiveness), while higher values are typically used for classification tasks (pattern recognition or statistical problems, where there is the need to analyse the flows in the correct order and without segmentation). FAN can also be used to perform offline experiments. In facts, with the proper time slot size and block timeout parameter values, FAN can analyse long-lived flows, even when they span over multiple fragments. The time slot size \(\tau\) is used for block grouping: a flow block \(\beta_i\) contains all the flows exchanged during time slot \(\tau\), of duration \(\tau\) (e.g. 60 seconds). The block timeout, instead, tells FAN to cache all the netflows of several blocks and to properly aggregate them to fix the effects of the active timeout. Such feature is needed by behavioural-based traffic detection engines (like [4]) since it enables time-coherent analysis. Further, it makes possible ex-post analysis, quality of service evaluations or the implementation of traffic classification algorithms. Since the active timeout value also affects FAN’s cache size, it should be set according with the Netflow probe configuration.

B. Plugins manager

FAN can dynamically load new plugins during its execution, since it exploits the programming interface to the dynamic linking loader, namely \textit{dlopen}, \textit{disym}, \textit{dlclose} functions. Each

\(^1\)Also known as IPFIX (RFC3955), Cisco Systems — http://tools.ietf.org/html/rfc3954
In order to evaluate the performance of the FAN framework, we implemented a set of plugins for anomaly detection: i) information theory based metrics (Entropy, Kullback Leibler and Rényi), ii) SYN-flood attack detection algorithm, iii) stock market and iv) k-means based anomaly detection. We used FAN to evaluate the plugins on a data set of real netflow gathered from a large Tier-II Autonomous System, but for space limitation, we only report the results for type i) metrics.

Figure 1 reports the experiments on around 2 hours of traffic that includes a publicly-declared DoS attack (Figure 1(a)).

FAN produced the results shown in Figure 1(b), evaluating 15 millions of flows with 6 plugins in around 17 minutes (i.e. 1011 seconds) of computations.

Figure 2 shows the effects of the plugin dependency subsystem optimisation in terms of execution time. For example, processing 10 millions of netflow the dependency subsystem saves 80 seconds of computations (around 25% of speed improvement), optimising 6 information theory based plugins.

IV. CONCLUSIONS AND FUTURE WORKS

In this work we present FAN, a framework for fast netflow analysis. It can perform online and offline evaluation for security and network monitoring purposes. FAN is modular and suitable for large scale networks, exploiting a rational plugin dependency subsystem for time execution optimisation. Furthermore, the timeout manager assures a time-coherent analysis of the flows, making FAN able to process them in the same order they cross the probe. As future work, we plan to implement plugins for netflow manipulation, like obfuscation or anonymization mechanisms.

Acknowledgments: this work has been partially supported by the TENACE PRIN Project (n. 20103P34XC) funded by the Italian Ministry of Education, University and Research.

REFERENCES