This material is based upon work supported by the Defense
Advanced Research Projects Agency (DARPA) under Contract No.
HR001119C0102. The opinions, findings, and conclusions stated
herein are those of the authors and do not necessarily reflect
those of DARPA.

pcap-ingestion

Introduction

The tools in this directory process pcap files to create CSV files and other output that is used as input for other analyses. When new pcaps are "ingested", they are parsed to extract key values that are used by other tools, and (optionally) preprocessed in several ways.

The central script in this directory is `process.sh`, which takes a parameter file as its single argument. (See `../../params` for information about the parameters, and `process.sh` for information about how those parameters are interpreted.)

The basic pcap ingestion pipeline has the steps described below. Note that some of these steps are optional, and may not be necessary for a given telescope or

application

1. (optional) *Pre-filter the input pcaps to remove unwanted packets*

   This step is optional, and there are other opportunities, later in the pipeline, for removing packets that should be excluded from the data. In some cases, however, it can be advantageous to remove packets that we know *a priori* we don't want to process as early in the pipeline as possible.

   For example, in our telescope, we have one /24 subnet that sees a very large amount of anomalous traffic. This subnet is the target of heavy DDoS attacks, and also appears to be the destination of a number of misconfigured services on the Internet. The result is that this small subnet sees a mix of packets that is *very* different from the mix any other subnet sees, and a volume of traffic that often dwarfs the total volume of the rest of the telescope. To prevent the behavior of this subnet overwhelming the rest of the telescope, our first step in our ingestion process is to filter out the packets addressed to this subnet (and set them aside, for a separate analysis).

   If the parameter `PCAPRAWDIR` is set, and not identical to `PCAPDIR`, it is assume that there is a pre-filtering step. The semantics of this step must be defined locally. See `./prefilter-pcap.sh` as an example implementation, and alter it according to your needs.

2. *Create CSV files from each pcap file*

   This step creates a CSV file for each pcap for which a CSV has not already been created.

   The format of the CSV files is documented in `../../C/cpcap2csv.c`.

3. (optional) *"Fix" the CSV files to start/end on hourly boundaries*

   We want each CSV file to begin/end on an hour boundary. We had an intermittent problem with our capture script, which would cause the pcap files (and the resulting CSV files) to begin and end at arbitrary times, and in some cases overlap (i.e. the last packets of the pcap file for hour H would have a timestamp later than the first packets of the pcap file for hour H+1).

   The `./refix-hours.sh` script "fixes" the CSV files by moving "out of place" CSV rows between the CSV files until each CSV file contains the data that corresponds to its expected start/end times.

   If the value of the `CSV3DIR` parameter is identical to the `FCSVDIR` parameter, then it is assumed that this step is not necessary, and is omitted.

4. *Create files for missing hours*

   If there was a failure during the capture (or any of pcap files are missing for any other reason), then it is possible that some hours have no data, and

therefore no CSV files are created for those hours. This can be awkward, because for some analyses it is convenient to assume that every hour has a corresponding CSV file. For example, if you want to analyze all of the data for the 24 hours following a given hour H, it's much easier to look at "the file for H and the next 23 hours" rather than to do date arithmetic to figure out what the correct month, day, and hour for each of those 23 hours.

The solution is to search for dates that don't have matching files, and create empty CSV files for those dates. This is done by `./fill-missing.sh`.

5. *Find the destination /24 subnets present in the data for each hour*

   Our telescope has changed over time, so that some subnets that were in the telescope are no longer present, and new subnets have been added. Therefore it's not always known *a priori* which subnets are active at a given time.

   Some analyses iterate over all of the destination subnets (usually /24 subnets), so it is useful to know what subnets are present in each hour. The `./find-subnets.sh` script scans the CSV file for each new hour and discovers which destination subnets are present.

6. (optional) *Compute some summary statistics*

   The `./trend-by-subnet.sh` script computes some summary statistics for the new CSV files, using the `firecracker` utility. This script also computes these summary statistics for packets sourced from any members of the set of acknowledged scanners and its complement (all of the sources that are *not* known to be acknowledged scanners).

   This gives a quick view of any large-scale emerging trends in the traffic (such as a large change in the frequency with which a specific destination port is scanned).