TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

institute of telecommunications

Exercise 2: Feature Extraction

Network Security 1 - Summer Semester 2019 (VU 389.159)

Communication Networks Group at the Institute of Telecommunications
T. Zseby, F. Iglesias, F. Meghdouri, M. Bachl, E. Zeraliu

1 Extracting Feature Vectors

Among other things, in Exercise 1 you learned to capture traffic and store it as pcaps. The pcap format has become a standard to capture and store network data, but it is not suitable for getting statistics and performing analysis because it only contains raw data. In this second exercise, you will learn to extract information from pcaps and prepare it for three types of analysis that consider network traffic from different perspectives.

1.1 Feature Extraction

Extracting information from raw data (pcaps in our case) to fit structured formats is commonly referred to as feature extraction (data is saved in CSV text or binary files). A feature is a dimension, variable or a quantity related to a network traffic instance. A traffic instance can be a packet, a flow, or network in a specific time span. Examples of network traffic features are: source/destination IP addresses, protocol, transport ports, flags, timestamps etc.

Feature extraction is a necessary step previous to the analysis. Moreover, most of the information contained in pcaps is not needed and by means of feature extraction we focus only on the information that is relevant for our application goals.

1.2 Tools

tshark is one of the most common tools for extracting packet information from pcaps. However, to extract flows or aggregated data we need additional tools. In this lab, we will use a new tool called Go-Flows[1]. This tool has a CLI (Command Line Interface) where the user specifies input pcaps, a configuration file, and extraction options. The tool outputs a csv file with the desired data. Go-Flows is installed in the lab computers. To make sure that the installation is fully operative, run the command `go-flows` in the Terminal.

[1] `Go-flows` is a tool for fast extraction of network flows developed by our group at TU Wien and not officially released yet. Ask the tutors for further information. An alternative to Go-Flows is using `Silk` and `Yaf`

1.3 Data

In this exercise we use the MAWI[2] network traffic data. MAWI is an acronym for the Measurement and Analysis on the WIDE Internet, or Measurement and Analysis of Wide-area Internet. The MAWI Working Group daily publishes 15 minutes of traffic from a network backbone data for research purposes. Data is open source and available at the organization homepage in addition to all information about the nature of the traffic and the main objectives. For security reasons, MAWI data is anonymized and does not contain payloads.

2 From pcap to packets

In this section we will extract information from each network packet separately. You will find a pcap file in your home directory: `Ex2_teamX.pcap`.

> **[rep-15]** Use `tcpdump` to extract information from pcaps and write in your report the data concerning the first 5 packets. Ensure that you show: time to live, identification, total length and options of IP packets. Also, don't convert addresses (i.e., host addresses, port numbers, etc.) to names. Print timestamps as seconds since January 1, 1970, 00:00:00, UTC, and fractions of a second since that time. Write in the report the syntax of the used command.

The next step is to learn how to use Go-Flows for extracting features instead of tcpdump. In a Terminal, change the working directory to your team working directory and run the command in listing 1.

Listing 1: Basic usage of the Go-flows exporter

```
1 go-flows run -perpacket features pcap2pkts.json
    export csv Ex2_teamXX.csv source libpcap
    Ex2_teamXX.pcap
```

Replace XX with your team number. This command calls the tool with an input configuration file (`pcap2pkts.json`), the option of exporting packets (`-perpacket`), the input pcap to

[2] `https://web.archive.org/web/20060615051547/http://www.wide.ad.jp/project/wg/mawi.html`

process, and the output csv in which to save the extracted information. The configuration file contains the list of desired features in addition to other specifications. Other Iana[3] features as well as personalized features are available (use `go-flows features` to get the whole list).

After extracting the csv, use a tool of your choice (Matlab, Python, R, Excel...) to answer the following questions in your report:
[rep-16] As for packet extraction, do you think that GoFlows has any advantage compared with tcpdump? If so, explain it.
[rep-17] What are the top 3 most used protocols in the pcap? how many packet were sent using these protocols? what is the percentage of each protocol with respect to the total number of packets?
[rep-18] How many packets (in proportion) are related to web traffic (HTTP, HTTPS)? And DNS traffic? Describe in the report the steps that you followed to obtain your results.

3 From pcap to flow vectors

The definition of a traffic flow given by IPFIX is extremely flexible. A flow is defined as a set of packets or frames passing an Observation Point in the network during a certain time interval. All packets belonging to a particular Flow have a set of common properties[4], which can vary depending on the use case. For the last three decades flows have been principally defined with the 5-tuple key: [IP source, IP destination, source Port, destination Port, Protocol], which states the communication for a specific application between endpoints, e.g., a TCP connection. However, the use of the 5-tuple is not justified in terms of security, it is simply a reminiscence from network policies implemented in the 1990s that have become a standard.

In this exercise we are going to use the 1-tuple AGM vector format. The AGM vector is originally proposed for the exploration and discovery of patterns in the Internet Background Radiation [5]. Defined for unidirectional traffic, this vector captures the behaviour of hosts by observing the use of eight lightweight flow features plus the total number of packets. Features are: srcIP, destIP, srcPort, dstPort, Protocol, TCPflag, TTL and pktLength. After selecting how hosts are profiled—either as data-senders (sources) or data-receivers (destinations)—and an observation time, the *basic AGM vector* stores the number of unique values, the mode, and the number of packets assigned to the mode of every one of the previous features (for instance, "dstPort" becomes "#dstPort", "mode_dstPort" and "pkts_mode_dstPort"). Thus, the basic AGM vector contains 22 features (7×3 + total number of packets; see Table 1).

Extract flows from your pcap. For this use the previous Go-Flows command without the `-perpacket` option and with the configuration file `pcap2flows.json`, which contains the description of the AGM format. Save the resulting CSV file as `Ex2flows_teamXX.csv` (you will need the same file for later exercises). Use a tool of your choice (Matlab, Python, R, Excel...) to answer the following questions in your report:
[rep-19] What is the percentage of IP sources that use: (a) only 1 protocol, (b) two protocols, (c) 3 or more protocols?
[rep-20] What is the percentage of IP sources that: (a) communicate with only 1 destination, (b) communicate with more than 10 destinations? (c) Can you mention at least two types of traffic that match case (b)?
[rep-21] Plot a histogram with the distribution of number of destinations contacted by each source and add it to the report (we recommend using MATLAB, but feel free to use other tools). Try to explain what you see in the histogram. Explain the followed steps and the configuration options that you used for plotting the histogram.

Table 1: AGM vector, format used in the experiments.

Flow key: srcIPAddress. **Observation window:** 10 seconds. **Feature vector:** flowStartSeconds, sourceIPAddress, distinct(destinationIPAddress), mode(destinationIPAddress), modeCount(destinationIPAddress), distinct(sourceTransportPort), mode(sourceTransportPort), modeCount(sourceTransportPort), distinct(destinationTransportPort), mode(destinationTransportPort), modeCount(destinationTransportPort), distinct(protocolIdentifier), mode(protocolIdentifier), modeCount(protocolIdentifier), distinct(ipTTL), mode(ipTTL), modeCount(ipTTL), distinct(tcpFlags), mode(tcpFlags), modeCount(tcpFlags), distinct(octetTotalCount), mode(octetTotalCount), modeCount(octetTotalCount), packetTotalCount

4 From pcap to aggregated vectors

In this last exercise, we want to extract the behaviour of the network as a whole from a temporal perspective. To do that we will extract time series from our pcaps by establishing a time window of one second. Our time series will show aggregated information corresponding to: three selected features: number of packets, number of unique sources, and number of bytes.

Use Go-Flows similarely to the previous exercise (without the `-perpacket` option) with the configuration file `pcap2aggFlows.json` to extract the new data.

[rep-22] Write in the report the mean, median and standard deviation values of each time series.
[rep-23] Plot the three time series together in the same figure (we recommend using MATLAB, but feel free to use other tools). Add the mean, median and standard deviation of each feature in the same plot.
[rep-24] Comment briefly whatever you find interesting from the plots and results. Are central tendency values (mean, median) representative?

[3] https://www.iana.org/
[4] https://www.ietf.org/rfc/rfc7011.txt.
[5] Iglesias, Felix, and Tanja Zseby. "Pattern Discovery in Internet Background Radiation." IEEE Transactions on Big Data (2017).