TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

institute of telecommunications

Exercise 3: Offline Analysis

Network Security - Advanced Topics (VU 389.160), Winter Semester 2018/2019

Communication Networks Group at the Institute of Telecommunications

3 Exercise 3: Offline Analysis

Your effort in decoding the covert channels has confirmed that indeed some data leakage operations are taking place within the Ministry at this moment. However, the arrested person seems not to be talking too much. Moreover, the machine from where the information was sent belongs to a bachelor trainee who assures that he does not know anything; actually, he claims that his computer must have been hacked.

In your opinion, the decoded messages found in the notebook clearly show that there is a problem within the minister's office network. Again, you suspect that covert channels are used. After reporting your conjecture to your superiors, they urge you to continue your investigation.

While the security staff physically checks the rooms of the minister, you quickly go back to your office and ask the network operators for recent traffic captures of the minister's office network. Fortunately, they do have them. Together with your team, you proceed searching covert communications that could have past undetected so far by the ministry's security barriers. Your main goal is to reveal the sender and receiver of a possible covert communication. Therefore, you must discover which traffic feature is used to conceal data, and, if possible, what information is clandestinely sent[1].

Now it is time to apply the methods and tools you have learned in Exercises 1 and 2. You can find the captured traffic as a pcap file to analyze in your `workfiles` folder (`/home/team??/workfiles/team??_ex3.pcap`).

Rep:3.a – Steps

Find and decode the covert channel. In the report, depict briefly every step so that your exploration can be successfully understood and reproduced. For every step, provide sound arguments and reasoning that support your decisions. In addition, specify all the information that characterizes the covert channel.

Note: Also, briefly comment on wrong attempts or approaches which did not lead to any improvement but allowed you to progress in your reasoning. Be short, accurate, and organized.

Important: Be sure that you have read the entire exercise sheet before you start solving this task (otherwise you may run into troubles).

[1]Since you move from the Ministry's test laboratory to the minister's office network, you expect to see different IP addresses

Rep:3.b – Message

Write in the report the message contained in the discovered covert channel.

You remember some tools that a colleague recently told you about during a coffee break. These tools might be useful to seize covert channels and discriminate irrelevant data and connections. Thus, you start reading about them:

3.1 IP address flow information

Theory Aggregated data of traffic flows can be useful to identify or filter suspicious IP addresses. A traffic 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*[1]. For this exercise, let us consider that a flow contains all packets going from one source address to one destination address. For instance, a TCP connection between client A and server B would be expressed with two flows: A to B and B to A.

Usage In the `workfiles` folder you can find a script `connections_of_ip.py`. This script takes the following arguments: a) a CSV file to analyze and b) a specific "source" or "destination" IP address. The script outputs some information about the activity of the provided "source" or "destination" and its main flow (in terms of number of packets).

Example Consider an hypothetical CSV traffic capture file called `example.csv`. The command

`%python ./workfiles/connections_of_ip.py --input example.csv --destination-ip 121.39.55.24`
outputs the following information:

```
Getting information for Destination : 121.39.55.24
Number of packets: 6
Number of connecting partners: 4
Most connections: 200.133.27.44
Number of packets for 200.133.27.44: 3
```

Results show that the host 121.39.55.24 has received only 6 packets from 4 different hosts, and that the source address 200.133.27.44 sent the most packets (3) to the selected host. With so few packets from a single source, it is unlikely that the host 121.39.55.24 has received covert information in the analyzed capture. The situation changes for this second example[2], however:

`%python ./workfiles/connections_of_ip.py --input example.csv --source-ip 110.23.9.32`

```
Getting information for Source : 110.23.9.32
Number of packets: 645
Number of connecting partners: 2
Most connections: 128.34.112.110
Number of packets for 128.34.112.110: 620
```

Here, the selected host has sent a considerable number of packets (645) to two different hosts, mostly to 128.34.112.110 $(620$ packets). Of course, this is not by itself suspicious to conceal a covert communication since it can be a perfectly normal scenario. Nevertheless, the introduced tool can be helpful to quickly display some relevant flow information about a specific "source" or "destination" under observation.

3.2 Multimodality estimation

Theory A suitable way of detecting possible covert channels is estimating the number of principal symbols $M(X)$ (multimodality estimation [2]). A rough estimation can be obtained as follows:

$$M(X) = \frac{\sum_i {n_i}^2}{\max\left({n_i}^2\right)} \tag{1}$$

where $M(X)$ stands for the number of principal symbols of the observed histogram X. X represents the relative frequencies of occurrences of different states of a phenomenon (i.e. a specific traffic-flow feature). n_i refers to the number of occurrences of a particular state i. Terms are squared to emphasize dominant symbols and fade less frequent symbols.

Example This method can be understood more clearly with an example. Assume we want to know whether a specific IP address *A* is using the packet length *L* to clandestinely convey information to destination address *B*. We filter the traffic and analyze the histogram of the packet length values used in the A→B flow (Figure 1). Based on the displayed histogram, a 2-symbol covert channel with two modes of length 107 and 671 could be in use. For this reason, we can consider this connection suspicious. The same information is provided by equation 1 without having to visually check the histogram. For the displayed case, out of 130 packets, length occurrences are:

[2]Note that, in the first example, we have selected to check the host in its role as a destination (`--destination-ip`), and in the second example as a source (`--source-ip`).

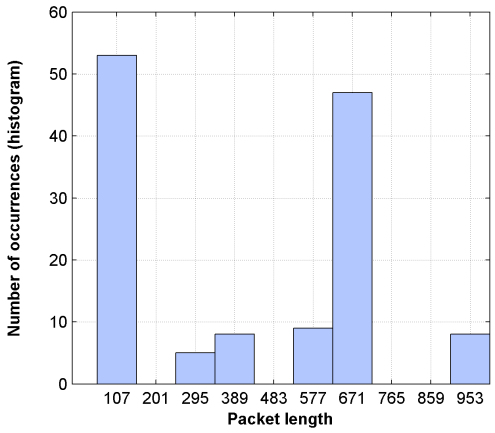

Figure 1. Histogram of packet lengths for an example case

107 $(53$ times), 295 (5), 389 (8), 577 (9), 671 (47) and 953 (8). The number of principal symbols can be estimated by

$$M(L)|_{A,B} = \frac{53^2 + 5^2 + 8^2 + 9^2 + 47^2 + 8^2}{53^2} \simeq 1.8 \tag{2}$$

The result ($\simeq 1.8$) gives an approximation on the number of principal symbols, i.e. 2.

Usage In the `workfiles` folder you find a script `multimodality.py`, which takes the following arguments: a) a CSV file to analyze, b) a feature to check, and c) a selection between "sources" or "destinations". It outputs a list, showing the following information:

X.X.X.X_1, No. of packets$_1$, No. of states$_{1,f}$, $M_1(f)$

X.X.X.X_2, No. of packets$_2$, No. of states$_{2,f}$, $M_2(f)$

...

where *X.X.X.*X_i is the source or destination address i and *No. of packets*$_i$ refers to the number of packets sent or received by i. *No. of states*$_{i,f}$ stands for the total number of observed states of feature f used by i, and $M_i(f)$ is the estimation of the number of principal symbols for feature f used by i. For the example above, the output would look like:

A, 150, 6, 1.8

This means that A sends 150 packets, from which the majority is shared between 2 out of 6 observed states.

This technique can be used to filter suspicious sources and destinations by observing the relative frequency occurrences of different features; mainly for those features where normal or random distributions are expected (i.e. where multimodal distributions are rare). Also $M(X)$ values close to 1 usually indicate the absence of a covert channel (since only a single dominant symbol hardly allows efficient communication). **Important!** There are multiple situations in which this detection scheme may fail. For example, when the analyzed time scope is too large compared to the time during which the covert channel is used (then symbols used for hidden communication become

less dominant), also when code symbols correspond to feature ranges instead of specific values, or when there is a significant amount of noise caused by non-related traffic.

3.3 Mean of autocorrelation coefficients

Theory In some cases, traffic hiding covert channels can be detected by looking at the time evolution of field values. For instance, some fields, such as the IP *Identification*, are expected to follow self-correlated patterns. Breaking such patterns can suggest the existence of a covert channel.

We define ρ_A as the average of a set of selected autocorrelation coefficients (absolute values) with different lags (e.g. $A = \{1, ..., 5\}$):

$$\rho_A = \frac{1}{N} \sum_{\tau}^{A} |R_\tau| \tag{3}$$

where N is the number of elements in A. R_τ is the autocorrelation coefficient of the time series $Y_1, ..., Y_z$ for the lag τ, defined in equation 4:

$$R_\tau = \frac{E[(Y_t - \mu)(Y_{t+\tau} - \mu)]}{\sigma^2} \tag{4}$$

with μ and σ^2 being the mean and variance of the time series, and E the expected value.

Usage In the `workfiles` folder you can find a script `autocorr.py`, which takes different arguments that identify a feature and a flow inside a CSV traffic file (run `autocorr.py` without any arguments for additional information). The script outputs a score $\in [0, ..., 1]$, which is the calculation of ρ_A. Values close to "0" means no correlation, whereas close to "1" indicates a high self-correlation.[3]

Example Imagine that host A sends 104 consecutive packets to destination B using two different TCP destination ports, 80 and 443. Packet by packet, the destination port sequence is as follows:

80,80,80,80,80,80,80,80,80,80,80,80,80,80,80,80,80,80,80,80,
80,80,80,80,80,80,80,80,80,80,80,80,80,80,433,433,433,433,
433,433,433,433,433,433,433,433,433,433,433,433,433,433,433,
433,433,433,433,433,433,433,433,433,433,433,433,433,433,433,
80,80,80,80,80,80,80,80,80,80,80,80,80,80,80,80,80,80,80,80,
80,80,80,80,80,80,80,80,433,433,433,433,433,433,433,433

`autocorr.py` obtains $\rho_A = 0.78$ for such sequence. It is not common in normal traffic to observe a highly overlapping destination ports (over long time periods) and, therefore, autocorrelation values are expected to be high.

In a second example, however, "Hello world!" has been binary encoded, using destination port 80 to mask '0' and destination port 443 to mask '1'. The 104-packet sequence in this second example is somewhat mixed and chaotic, showing a low $\rho_A = 0.09$:

80,433,433,80,433,80,80,80,80,433,433,80,80,433,80,433,80,433,
433,80,433,433,80,80,80,433,433,80,433,433,80,80,80,433,433,
80,433,433,433,433,80,80,433,80,80,80,80,80,433,433,433,
80,433,433,433,80,433,433,80,433,433,433,433,80,433,433,433,
80,80,433,80,80,433,433,80,433,433,80,80,80,433,433,80,80,
433,80,80,80,80,433,80,80,80,80,433,80,80,80,80,433,80,433,80

[3]Note that the flow must contain a sufficient number of packets in order to provide a meaningful, reliable estimation.

Important! Whether to expect high or low self-correlated sequences of values in a flow strongly depends on the traffic feature under analysis as well as the type of searched covert channel technique. Deep knowledge and understanding of traffic features and covert channels are mandatory for effective detection!

3.4 Combining tools and filters

This exercise requires the combined deployment of introduced tools and methods. Correct pre-processing and filtering is determinant to remove irrelevant data and to progress in the task of seizing the covert channel. The exploration entails repeatedly filtering, generating CSV files, analyzing CSV files, suggesting and testing hypothesis, and improving filtering criteria.

For example, LibreOffice spreadsheets can be a suitable tool for filtering data. Results from Wireshark, Rapidminer[4] and the `multimodality.py` script can be saved as CSV files and imported into a spreadsheet. Then, filters can be applied using: "Data>Filter>Standard Filter..." by selecting diverse criteria (Figures 2 and 3).

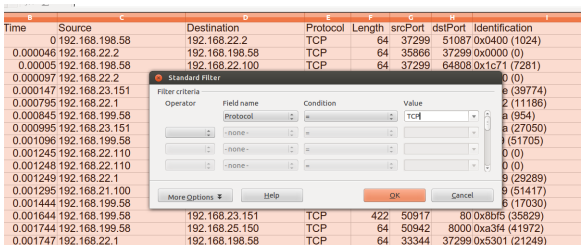

Figure 2. Filtering entries of a Wireshark CSV output file

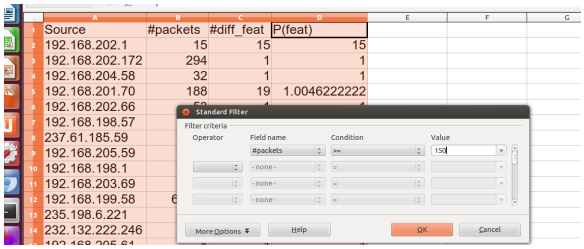

Figure 3. Filtering entries of an output file obtained from the multimodality script

References

[1] RFC 7011 - Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information. Technical report, Internet Engineering Task Force (IETF), Sept. 2013.

[2] B. W. Silverman. Using kernel density estimates to investigate multimodality. *J. Royal Stat. Soc. B*, 43(4):97–99, 1981.

[4]Have a look on the "Filter Examples" and "Write CSV" operators in Rapidminer.