DETECTION AND RECOGNITION OF PERIODIC COMMUNICATION IN NETWORK TRAFFIC

Josef Koumar

Faculty of Information Technology, CTU in Prague, Thrakurova 9, Prague 6, Czech Republic

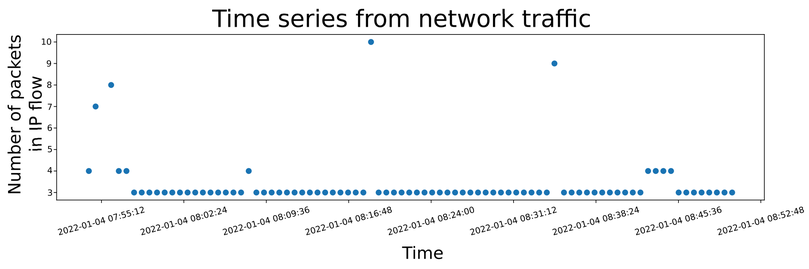
MOTIVATION

In (encrypted) network communication appears automatically generated communications that have periodicity behavior to update or check a system or application state. This work focuses on detecting this type of communication and its description to a set of attributes that we use to create a side channel of encrypted network communication, which allows us to obtain information about the process that generated the encrypted communication.

Examples of periodic communication include Facebook Messenger, MS Teams, network services, versioning systems, remote storages, email clients, antivirus clients, and often Command and Control (C&C) malware communications.

TIME SERIES FROM (ENCRYPTED) NETWORK COMMUNICATION

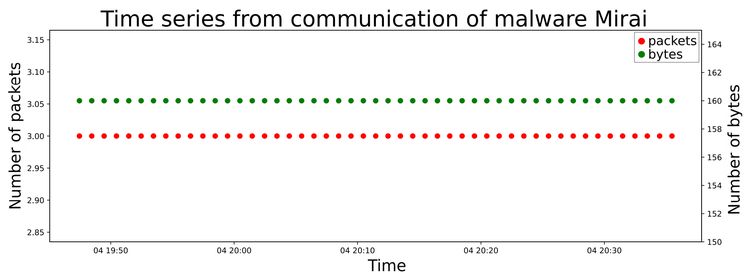
A time series from network traffic is a sequence of flow records for which the transfer time of data points meets the requirement: $t_{i-1} \leq t_i \leq t_{i+1}$. Almost all time series from network traffic are unevenly sampled and often contain spaces between data points that are much larger than usual spaces.

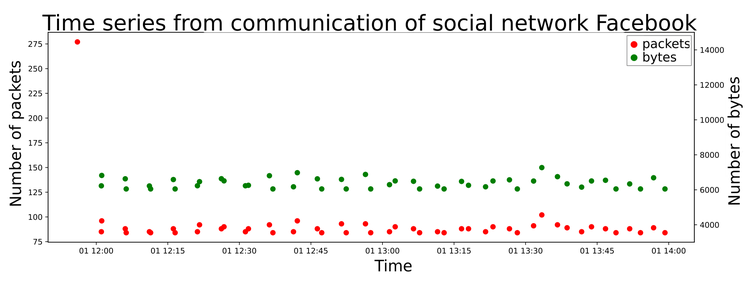


Network traffic is split into multiple time series by partition traffic by network dependencies. A network dependency is a relationship between server and client where the server provides some service to the client. Splitting traffic by network dependencies in most cases ensures that individual data points of the time series are generated by one process.

PERIODIC TIME SERIES FROM NETWORK COMMUNICATION

We define two periodicity behaviors in time series from network traffic. The first is periodically repeating the same IP flow multiple times. Second is periodicity behavior that is affected by carried data or user activity. Data points in this type of behavior are in some "small" interval.

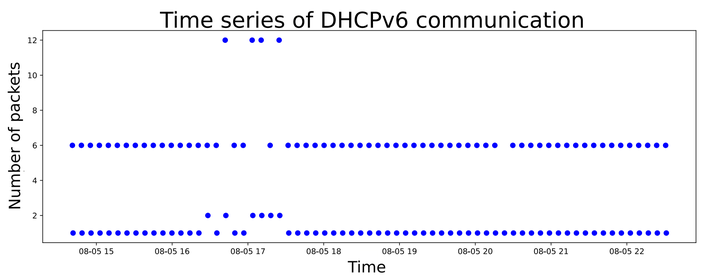




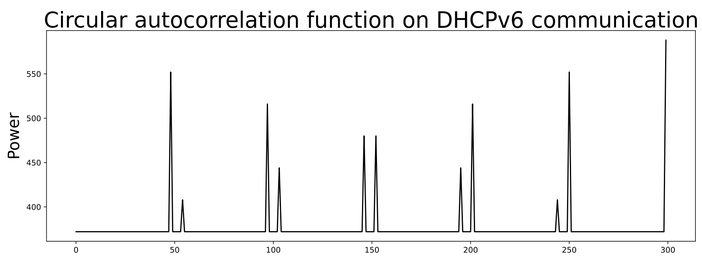
MODEL DETECTION PERIODICITY

The model for detection of periodicity behaviors in time series from network communication is based on the Lomb-Scargle periodogram (LS periodogram) that was selected based on the type of time series (unevenly sampled). We also use the Autocorrelation function to get candidates on period. Model diagram is placed in right down corner.

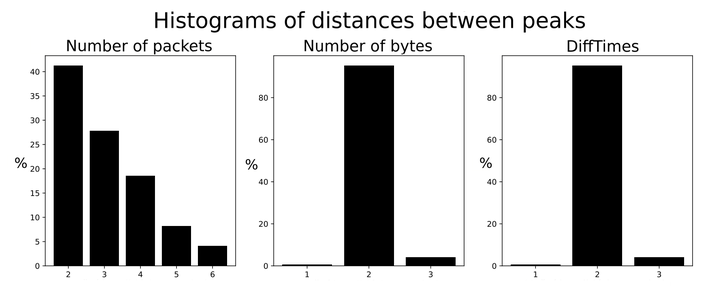
Example of using model for detection periodicity behavior on time series from communication of DHCPv6 is below.



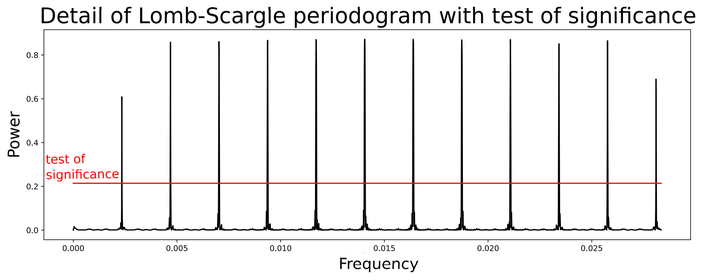
Then we apply Circular Autocorrelation function on each metric of the time series. Example for number of packets in IP flow:



We compute histogram of distances between of autocorrelation peaks. Mayor distances are candidates on periodicity. Candidate must be in each metric.



Then we apply the LS periodogram on the time series. If near the frequency of candidates from autocorrelation function is significance peak (tested with a test of significance) then the time series contains periodic behavior.



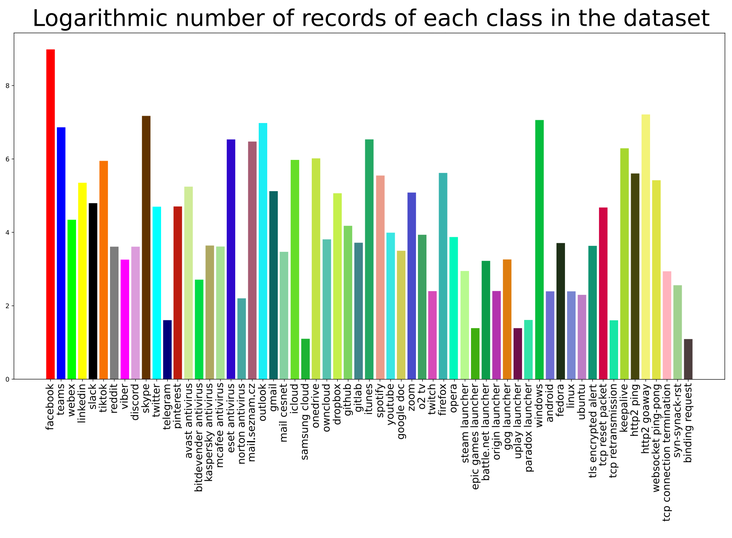
DATASET

We created a dataset of periodicity behaviors for applications, services, and operating systems. We use 1 tb of network traffic from multiple sources.

The processes that have periodicity behaviors:

- social networks (Facebook, MS teams, Slack, ...)
- remote storage (Google Drive, Github, ...)
- actualization of operating systems
- antivirus programs (Eset, Avast, Kaspersky, ...)
- game clients (Steam, Epic Games, ...)
- network services and protocols (Keep-alive, HTTP2 ping, DNS, ...)

The number of instances of each class in the dataset:



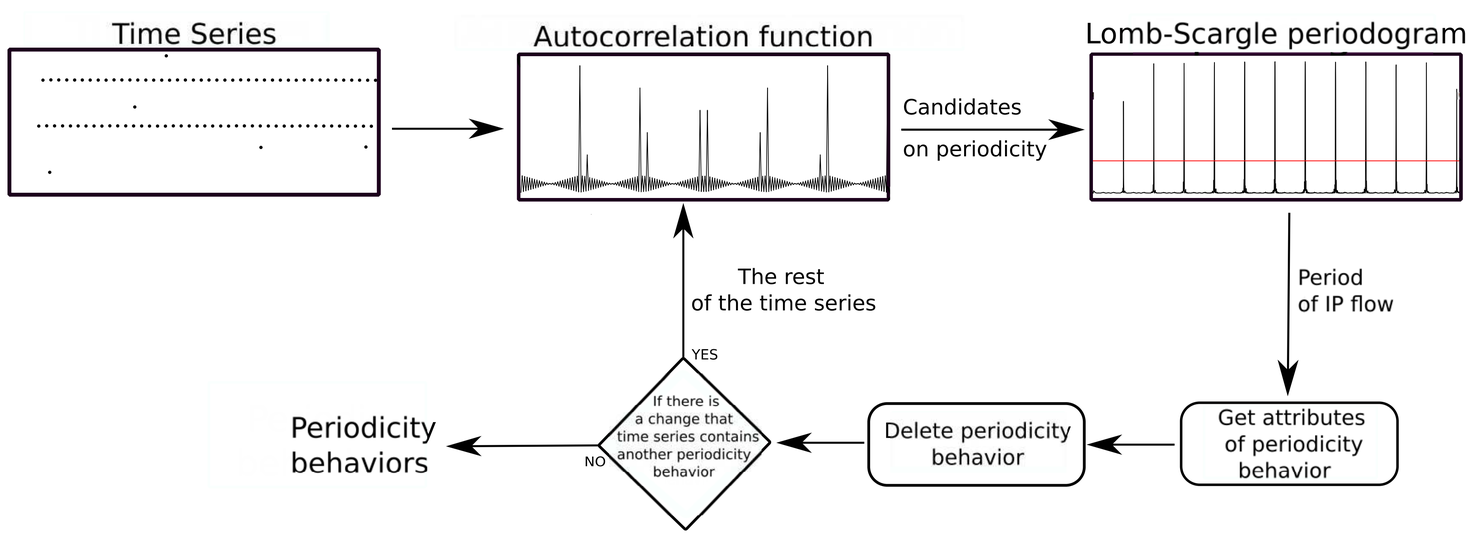
PERIODICITY CLASSIFIER

We trained machine learning models from which it turned out the best XGBoost algorithm with an accuracy of 90%. The Selected XGBoost model was validated by K-fold validation with an accuracy of 85%.

	Accuracy	Precision	Recall	F1-score
kNN	62	50	40	41
Decision Tree	77	45	46	45
Random Forest	89	87	64	72
XGBoost	90	82	66	71

CONCLUSION

We proved that the side channel of encrypted network communication based on periodicity behaviors of processes exists, and it is possible to use it for detection applications, services and operating systems by using machine learning models.



Contact: koumajos@fit.cvut.cz; koumar@cesnet.cz