Network security: Feature engineering using graph databases to improve anomaly detection model performance

Project Leader: Gordon Baggott
Project Helper: Neil Venables

Within Lloyds Banking Group Security Operations Centres (SOCs) undertake the correlation and analysis of security events to protect the organisation from external cyber-attacks. Predominantly, these data are activity logs sourced from many heterogenous systems using multiple and evolving communication protocols (e.g. TCP,DNS, HTTP, properties of transferred files, etc.).

Anomaly detection is the identification of items, events or observations which do not conform to an expected pattern or other items in a data set [1]. These detection approaches are commonly used across many industries to identify abnormal events, or patterns of events which are subsequently investigated to identify malicious activity.

Network intrusion is a common route for cyber criminals to gain access to an organisation's systems to undertake activity that will result in disruption, data theft and/or fraud. Networks are monitored with information collected in real time for elements such as IP Addresses, ports, communication protocols, number of connections made and bytes/type of data transferred. Over time these data present us with a large number of interactions between elements and amount to very high volumes with heterogenous properties across a single dataset.

A well-recognised approach to extracting features from network intrusion data is to build a graph database to capture security objects (nodes) and their semantic links (edges) [2, 3]. These approaches can be used more generally in wider cybersecurity data also [4]. From these graphs many algorithms can be used to create meaningful features that can be used to develop anomaly detection models – path properties, centrality properties, community properties etc. [5]

Given this you should attempt the following:

1. Take the pre-processed and labelled csv dataset we have provided and create a graph model. This data is a simplified version of the raw packet capture (PCAP) data found in the UNSW-NB15 Computer Security Dataset [6, 7].
2. Using your graph model derive a number of features that capture the topology of the graph well.
3. Test out your features using a supervised anomaly detection model to try and predict as many actual attacks as possible while reducing false positives.
4. Review your engineered features with reference to the anomaly model results and try to improve model performance by trying new or adjusted features.

We recommend using Neo4J Desktop edition to build then explore your graph and start to engineer features [8].

To test how well your engineered features are working we suggest you use a simple, supervised anomaly detection technique (e.g. K-nearest neighbour). When you test new features try to avoid tuning your model differently so you can assess the improvements due to the features changing rather than the model.

There are a number of useful anomaly detection algorithms available as part of the scikit-learn toolkit [9].

The focus of this project should be the algorithms used to create novel features from the graph database rather than the anomaly detection approach used.

We are hopeful that your research will contribute to the continuous evolution of Lloyds Banking Group's Cyber programme.

References:

[1] https://cumulocity.com/guides/machine-learning/anomaly-detection/
[2] Sec2graph: Network Attack Detection Based on Novelty Detection on Graph Structured Data (inria.fr)
[3] Temporal graph-based approach for behavioural entity classification 2105.04798.pdf (arxiv.org)
[4] A Practical Approach to Constructing a Knowledge Graph for Cybersecurity - ScienceDirect
[5] Neo4j Graph Data Science - Developer Guides
[6] The UNSW-NB15 Dataset | UNSW Research
[7] UNSW-NB15 Computer Security Dataset: Analysis through Visualization 2101.05067.pdf (arxiv.org)
[8] Neo4j Desktop Download | Free Graph Database Download
[9] 1.6. Nearest Neighbors — scikit-learn 1.0.2 documentation