Dynamic Ensembles for Anomaly Detection: An Application in Malicious Domain Investigations for US Army NETCOM

Lydia Barit, Kathryn Dula, Blake Jacobs, Harrison Leinweber, John McCormick, Roberts Nelson

1 Executive Summary

1.1 Project Overview and Discussion

The purpose of our project was to help NETCOM understand the effectiveness of anomaly detection models in use cases pertinent to the Command's mission and effectively incorporate these models into its current workflow. We accomplished this through an extension to NETCOM's existing Apache Airflow malicious URL detection pipeline. We developed an additional Directed Acyclic Graph to engineer features, run four anomaly detection algorithms (xStream, isolation forests, local outlier factor, and one-class spanning vector machine), and generate Shapley Additive Explanations for the data. We then present these results and explanations in a web-interface of our own design. Finally, we encapsulate our work within a sample risk framework to show how it can contribute to cyber risk management practices both on the front line and across the Command.

1.2 Project Deliverables

The following comprise our final set of deliverables:

1. **Project Report**: A document which details the datasets we analyzed, model selection rationale, our design process, and a sample risk framework.

2. **Final Project Presentation**: Audiovisual presentation delivered on 3 May 2022 which summarized this project report and provided an opportunity for stakeholders to seek clarification and further information.

3. **Project Video**: A condensed version of the project presentation is available here: `https://youtu.be/zBE4twWFRD8`

4. **User Interface Code**: Implemented in RShiny, our front end allows the analyst to interact with the data and models and provides the ability to see detailed anomaly score explanations, mute features, and mark specific URLs for further investigation. Directions for installation are provided with the user interface code.

5. **Amazon MWAA Cloud Pipeline**: A stand-alone Directed Acyclic Graph (DAG) and corresponding source code is provided to clean, score, and generate explanations for the example NETCOM NULLFOX data. The example data, intermediate results, and final data files are also provided for clarity in interpreting the source code.

1.3 Documentation

1.3.1 User Interface

Source code for the R-Shiny web application consists of a single R script "app.R" and an adjacent directory '/www', which contains the .jpg files used in the application's display. These have been included in a compressed zip file labeled "NETCOM_App_Prototype.zip" and are available on the projects github page.

The app script file has been documented throughout with in-line comments and headers. These headers have been structured in a outline format, which when loaded into R-Studio provides a clear organizational structure for reviewing, understanding and modifying the app. The code is split into four primary sections: Loading the Data from AWS (which includes some additional data processing), the UI, and the Server. Each of these sections additionally have helper functions which are essential to procedurally generating the app based on the features included in the data. These functions will be essential to maintaining the app moving forward, given the high likelihood that features may change as the pipeline is improved and tweaked. However, even with the procedural generated code, the app has strong coupling with the decision to include the four anomaly detection algorithms. Adding more algorithms (or removing some of the existing ones) will require significant refactoring.

Additionally, the app is currently connected to the AWS bucket created by the DAG scripts and the Amazon MWAA Cloud Pipeline. This can be modified by editing which aws s3 bucket is being referenced and changing the AWS authorization id and key. Though there is weak coupling with the specific data structure generated by the pipeline as designed, this can be changed if desired by refactoring the data processing portion of the app.

Finally, regarding the deployment of the app, a number of options are available for deploying an R-Shiny app. The easiest method is deploying the app to shinyio, a free service provided by RStudio. This service can be upgraded to a premium tier which provides additional resources and bandwidth dedicated to the app. Additionally, a third party or enterprise hosted R Shiny server could be established. This capability is provided by the AI2C's development platform Coeus.

1.3.2 Amazon MWAA Cloud Pipeline

Source code for a stand-alone DAG in Amazon Managed Workflows for Apache Airflow (MWAA) is provided along with example data that was produced by the DAG. This DAG was tested on Amazon MWAA to ensure that the requirements file and directory structure are correct and that the DAG can run without error. The code and data is provided as a compressed zip file with the name "NETCOM_cloud_backend.zip". Inside this directory are two subdirectories: "dags" and "data".

The "dags" directory contains all the source code for running the DAG. In Amazon MWAA you should set the root of the DAG to this directory. The directory has a "requirements.txt" file that AWS MWAA reads to manage the DAG dependencies. The DAG itself is provided in "anomaly_detection_dag.py". The source code that the DAG calls is located in the "src/" directory. The source code includes "CleanEnrichedData.py", "CreateAnomalyScores.py", and "CreateSHAP.py" which handle the data cleaning, scoring, and SHAP explanations respectively. Additionally, the directory includes the source code for the python implementation of xStream which

you can also find on Github.[1]

The "data" directory contains the example NULLFOX output data (enriched_altIP.csv), the intermediate results in the pipeline (CleanEnrichedData.csv), as well as the final outputs of the pipeline. The pipeline produces five final outputs. The first is "enriched_altIP_with_scores.csv" which is a copy of the original inputs to the pipeline with the four anomaly scores added to the end of the data. The other four outputs are SHAP values for each of the four anomaly detection methods. These files are intended to be used as concrete examples of the inputs and outputs of the DAG as it progresses to clarify the comments and documentation provided in the source code.

In order to get integrate the DAG into NETCOM's NULLFOX workflow, the only changes that should be needed are some resource strings for Amazon S3 buckets and object keys. All locations in the code where these changes need to happen are marked with a "TODO" comment to aid in the transition to NETCOM's production environment.

1.4 Suggestions for Future Work

We suggest the following five work streams for future work:

- Searching for "malicious URLs" in an unlabeled data set leads to difficulties in evaluating model performance. Websites can present numerous threats to the enterprise (ex. spam, phishing, drive-by downloads, data exfiltration, or general espionage). We suggest determining the specific threat that NETCOM is searching for in order to allow for better model evaluation and comparison.

- Not all anomalous URLs are malicious. Further research is needed to determine whether anomaly detection models are the best paradigm for investigating malicious URLs. We suggest investigating additional families of algorithms to see if there exists an alternative which performs better on this data.

- We suggest increasing the window of observation from one week. Evaluating performance and noise on a well-labeled dataset over periods ranging from bi-weekly, monthly, bi-monthly, and semi-annually would be ideal.

- Further work is needed in order to enable our pipeline to work in-stream and on a dynamic dataset.

- We suggest utilizing our sample risk framework to create and maintain an organization-specific risk register which provides analysts and managers the details needed to map *anomaly* scores to *risk* scores. This register would allow for effective and efficient prioritization and response to findings.

[1] https://github.com/cmuxstream/cmuxstream-core/tree/master/python