Code Inspection Report

Team: Lighthouse Solutions

Version: 1.0

Date: 2/21/2023

Client: Ben Drozdenko of the Naval Undersea Warfare Center Division Newport (NUWCDIVNPT)

Prepared by Johnny Driscoll, Sean Staton, Dylan Haughton, Brody Looney and Andy Howe

Abnormal Network Traffic Flow Dashboard Tool

Code Inspection Review

Table of Contents

1. Introduction

This is a capstone project for Ben Drozdenko representing the Naval Undersea Warfare Center Division Newport (NUWCDIVNPT), in partial fulfillment of the Computer Science BS degree for the University of Maine. The purpose of this project is to give the client a viable means of analyzing traffic that moves around their network in order to protect the integrity of confidentiality of all data that flows through. The project serves to detect statistical anomalies and give minimalist, user-friendly reports.

1.1 Purpose of This Document

This document naturally serves to outline the code inspection report for the *Abnormal Network Traffic Flow Dashboard Tool* that is being developed for Ben Drozdenko of the Naval Undersea Warfare Center Division Newport (NUWCDIVNPT). In this document there will be detailed information about the coding and commenting conventions used in this project, the defect checklist used during our inspections, the overall structure of the code inspection process that was implemented, the modules of the code inspected, and the defects that were discovered during the inspection.

1.1a References

Looney, Brody, et al. *Abnormal Network Traffic Dashboard Tool System Design Document.* 31 Oct. 2022.

Looney, Brody, et al. *Abnormal Network Traffic Dashboard Tool System Requirements Specification.* 06 Oct. 2022.

Looney, Brody, et al. *Abnormal Network Traffic Dashboard Tool User Interface Design Document.* 28 Nov. 2022.

Looney, Brody, et al. *Abnormal Network Traffic Dashboard Tool Critical Design Document.* 13 Dec. 2022.

Moustafa, Nour, and Jill Slay. "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)." *Military Communications and Information Systems Conference (MilCIS)*, 2015. IEEE, 2015.

Moustafa, Nour, and Jill Slay. "The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 dataset and the comparison with the KDD99 dataset." *Information Security Journal: A Global Perspective* (2016): 1-14.

Moustafa, Nour, et al. "Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks." *IEEE Transactions on Big Data (2017)*.

Moustafa, Nour, et al. "Big data analytics for intrusion detection system: statistical decision-making using finite dirichlet mixture models." *Data Analytics and Decision Support for Cybersecurity. Springer, Cham, 2017*. 127-156.

Sarhan, Mohanad, Siamak Layeghy, Nour Moustafa, and Marius Portmann. NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems. In *Big Data Technologies and Applications: 10th EAI International Conference, BDTA 2020, and 13th EAI International Conference on Wireless Internet, WiCON 2020, Virtual Event, December 11, 2020, Proceedings* (p. 117). Springer Nature.

1.2 Coding and Commenting Conventions

The majority of our development project was coded in Python, due to this our development team decided that it would be best to follow PEP 8 styling. PEP 8 provided us with a solid structure when concerning our code styling. By following PEP 8 we were able to keep our code very consistent and readable across the three different areas of the project.

A summary of the conventions that were often used from PEP 8 in our project are as follows; four spaces per indentation level, a maximum line length of 79 characters, blank lines between definitions, imports are on their own line, one space before and after operators, single line comments start with a # and a space between the first character, docstrings are contained within triple quotes, snakecase is used for variable names, and lastly variable names do not exceed 20 characters. We also added in a few conventions that were not stated in PEP 8 guidelines. These conventions included adding comments often and keeping this comments updated and accurate. We also included a convention of ensuring that variable names were clear and accurately described their purpose. Lastly, we included a convention that there would not be any unnecessary whitespace in the code produced.

PEP 8 Reference:

van Rossum, Guido, et al. *PEP 8 – Style Guide for Python Code*, 2013, https://peps.python.org/pep-0008/.

1.3 Defect Checklist

This section contains a table that lists the possible defects that could be contained in the code base. The chart contains possible defects under certain categories which label the type of defect. There is then a defect classification for each defect to show the impact that the defect would have on the system if it were present.

Defect	Defect Classification		
	Low Impact	**Medium Impact**	**High Impact**
A. Code Styling			
1. Snakecase with lowercase letters not used.	**X**		
2. Unnecessary extra Whitespace	**X**		
3. No space between operators	**X**		
4. Code is not appropriately sectioned out	**X**		
5. Variable names provide an unclear description	**X**		
6. Variable names are too long (> 20 characters)	**X**		
7. Line length is too long (> 79 characters)	**X**		
8. Imports are not on separate lines	**X**		
B. Commenting Conventions			
1. Single line comments do not start with #	**X**		
2. Docstrings are not enclosed in triple quotes	**X**		
3. Comments are not used often	**X**		
4. Comments do not provide accurate information	**X**		
C. Logical Errors			
1. Incorrect variable name used		**X**	

2. Incorrect indentation (not 4 spaces)	**X**		
3. Unused variables	**X**		
4. Unreferenced functions or methods	**X**		
5. Unnecessary function or method calls		**X**	
6. Incorrect data types are used		**X**	
D. UI Requirements			
1. There is no functionality implemented to start the capture of data.			**X**
2. There is no functionality implemented to stop the capture of data.			**X**
3. There is no functionality implemented to swap between training and capturing modes.		**X**	
4. There is no functionality implemented to upload a file to train the machine learning model.			**X**
5. There is no functionality to allow for a user to choose whether data is abnormal or not.			**X**
6. There is no screen implemented that displays individual cases of abnormal network traffic flow.		**X**	
E. Machine Learning Model Requirements			
1. The model does not provide predictions for each line of data being processed			**X**
2. The model does not have the functionality to be trained from a file given to the model			**X**
3. The model does not have the functionality to be trained from live data		**X**	
F. Database Requirements			

1. Database is missing a table that was stated in the requirements			**X**
2. A table is missing a column that was stated in the requirements			**X**

Figure 1. Defect Checklist. This table shows the checklist that was used to detect defects in our codebase during the code inspection. Each defect is connected to a certain type of defect and each has a rating for the impact of the presence of the defect.

2. Code Inspection Process

So far, we have completed one code inspection - of all three departments - and one Zeek demonstration. The schema for the inspection outlined below is quick, efficient, and provides effective / necessary input for code improvement. Violations of the checklist were discovered in the preliminary inspection, but they were not much more than mere code-styling issues.

Our goals on the code, itself, are being met with regards to our time constraints. The majority of department responsibilities have been met in isolation - the UI, machine learning algorithm, database, and Zeek all successfully operate by themselves. These departments still need a few more features to be implemented and they need to be integrated with one another. Code inspections so far have considered only those features in isolation.

2.1 Description

Our code inspection process is typical and quasi-standard. Three inspections occur officially, with an informal procedural inspection as a secondary affair. These three official inspections concern the UI code, machine learning code, and the database code. Five roles are delegated, one for each member, with the exception of the UI inspection which will be elaborated upon; Author, the author of the code, Reader, who explains and dissects the code, Inspector, who checks the code against the above checklist, Recorder, who records the complaints of the inspector, and Moderator, who facilitates the entire process.

These roles rotate, such that everyone may get a new role at some point. The UI inspection, since there are two authors, requires one individual to take two roles - the roles often paired are Moderator and Recorder, as they are the least biased on the topic at hand. The informal procedural inspection concerns the use and implementation of ZeekFlow, where updates on that process are demonstrated, though there is no code to show on the matter.

2.2 Impressions of the Process

Our group's code inspection report went quite well, we were able to identify and fix several white space and variable name errors. We were also able to identify and remove some unnecessary code. Overall our code inspection report was a success and we were able to fix some potential issues. If we were to go through this process again, conducting the CIR in person would have been more beneficial as it would've made taking notes while inspecting the code easier.
The best modular unit of our program would be our machine learning algorithm. All that needs to be fixed in this module is some general commenting and changing of variable names to match snake_case. The worst modular unit of our program would be the UI. This is because we have not implemented our train from file method, or our start capture method. Although this will be resolved when all portions of our program are implemented together.

2.3 Inspection Meetings

This table below displays the different sections of our code inspection meeting. Each section displays information such as the data and time, the participants and their respective roles, and lastly the code that was reviewed during that specific section.

Report	**Date**	**Location**	**Time Started**	**Time Ended**	**Participants**	**Roles**	**Code Considered**
Code Inspection I	21 Feb, 2023	Virtual	6:00PM	6:25PM	All	Author: A. Howe, S. Staton Moderator: J. Driscoll Reader: B. Looney Recorder: J. Driscoll Inspector: D. Haughton	UI (All lines)
Code Inspection I	21 Feb, 2023	Virtual	6:25PM	6:50PM	All	Author: B. Looney Moderator: J. Driscoll Reader: D. Haughton Recorder: A. Howe Inspector: S. Staton	Machine Learning Model (All lines)
Code Inspection I	21 Feb, 2023	Virtual	6:50PM	7:15PM	All	Author: J. Driscoll Moderator: D. Haughton Reader: S. Staton Recorder: A. Howe Inspector: B. Looney	Database (All lines)
Code Inspection I	21 Feb, 2023	Virtual	7:15PM	7:30PM	All	Reader: D. Haughton	Zeek (All lines)

Figure 2. Code Inspection Schedule. This displays the code review meetings that were held along with the information pertaining to each meeting.

3. Modules Inspected

The table below describes each code module that was inspected during the code inspection. There is a brief description of each module along with the planned implementation versus the actual implementation. There is also a column that describes the trade offs made.

Code Module	**Description**	**Planned vs actual implementation**	**Trade Offs**
Machine learning algorithm	The machine learning algorithm or model, will take in live capture data from zeek flow, and then determine whether it is “abnormal” or not based on its training.	There are no major changes or adjustments with the machine learning aspect, and the .csv file and testing that is being used remains the same.	N/A
User Interface	The user interface consists of a large “graph” that takes up the high majority of the screen, followed by a few buttons: one to start capture and one to end, and also a place to enter the settings of the ML model.	Although mentioned within the SDD exact implementation was not discussed, and so the only implementation change we made was adding the machine learning models to the UI and making the graph bigger.	● Increases communication between modules ● Harder to test and implement ● Increases functionality
Database	The SQL database communicates with both the UI	The only major change was making the database also be	From an architectural standpoint this approach made more logical sense. Only

	and the ML model and will store the model's settings as well as all relevant traffic flow data coming from the ML model (Section 3.1 of SDD)	able to communicate with the user interface as well as the ML model. Secondly there were additional tables added in the database: one for storing the machine learning settings, and one for storing the IDs of network traffic that when above our threshold value	possible downside is more communication from the database so if there will be more SQL database calls within our tool.

Figure 3. Modules of the Codebase. This provides information on the different modules in the systems codebase.

4. Defects

This table below lists the defects that were found in each module of the codebase. The table displays the module and line where the defect occurred. It also provides the code which relates to the information in Figure 1 along with a description of the defect. Lastly, the category of the defect is shown.

Module where the defect lives	**Defect code and description**	**Category**
UI		
Line 13: Does not follow naming conventions (not using snakecase)	(A. 1) Line does not use the specified format. Reducing overall synchronicity.	Naming Convention
Line 14: Does not follow naming conventions (no snakecase)	(A. 1) Line does not use the specified format. Reducing overall synchronicity.	Naming Convention

Line 14: Snakecase not used on imported functions (cannot be changed)	Line does not use the specified format. Reducing overall synchronicity.	Naming Convention
Line 56: Remove unnecessary comment	(B. 4) Comment wasn't sufficient enough to stay in.	Commenting Convention
Line 66: Remove unnecessary comment	(B. 4) Comment wasn't sufficient enough to stay in.	Commenting Convention
Lines 64-65: Add comments (need explanation on plotting methodology)	(B. 3) Line was complex enough to warrant a comment.	Add Comment
Unit Missing in Module: There is no button to select to change between training and capturing modes. There is also no functionality implemented to allow for this behavior to occur.	(D. 3) There is no functionality implemented to swap between training and capturing modes.	Missing Functionality
Unit Missing in Module: There is no area that displays network traffic flow and there is no button to select that displays this	(D. 6) There is no screen implemented that displays individual cases of abnormal network traffic flow.	Missing Functionality
Machine Learning		
Need comments for imports	(B. 3) Line was complex enough to warrant a comment.	Add Comment
Lines 2-9: Need comments	(B. 3) Line was complex enough to warrant a comment.	Add Comment
Section 3, lines 1 and 2: Comments needed	(B. 3) Line was complex enough to warrant a comment.	Add Comment

X TRAIN and X TEX need to be snakecase	(A. 1) Line does not use the specified format. Reducing overall synchronicity.	Naming Convention
Section 8, line 1: Shorten	(A. 7) Line was too long and needed to be shortened for clarity of reading.	Shorten Length
Sections 1-8: need comments	(B. 3) Line was complex enough to warrant a comment.	Add Comment
Sec 9-11: needs place holder comments	(B. 3) Line was complex enough to warrant a comment.	Add Comment
Unit Missing in Module: The model is unable to be trained form live data currently (This unit will be implemented during integration)	(E. 3) The model does not have the functionality to be trained from live data	Missing Functionality
Database		
Add var description comment	(B. 3) Line was complex enough to warrant a comment.	Add Comment
Line 10: Table name needs to be snake case	(A. 1) Line does not use the specified format. Reducing overall synchronicity.	Naming Convention
Lines 17, 29 and 53: Make ID lowercase to follow convention	(A. 1) Line does not use the specified format. Reducing overall synchronicity.	Naming Convention
Lines 38,39,44 and 45: Unnecessary function call (redundant)	(B. 5) Function call was inserted twice. Making it redundant.	Redundancy
Lines 6-42: Unnecessary whitespace	(A. 2) Excess spacing in code.	Cleanliness

Between line 56 and 57: Improper spacing	(A. 2) Excess spacing in code.	Cleanliness
Lines 72-74: Excess whitespace	(A. 2) Excess spacing in code.	Cleanliness
Lines 89-91: Excess whitespace	(A. 2) Excess spacing in code.	Cleanliness

Figure 4. Codebase Defects. This shows the defects that appeared in the codebase during the inspection and provides information for each defect.

Appendix A – Agreement Between Customer and Contractor

This document states the code inspection review process and results for the *Abnormal Network Traffic Flow Dashboard Tool* contracted by Ben Drozdenko and the Naval Undersea Warfare Center Division Newport (NUWCDIVNPT). Both the customer and the development team agree that this document clearly states the process and outcome of the code review.

In the event that there are future changes made to this document the development team and customer will discuss the changes during our bi-weekly check-in meetings. If everyone agrees on the changes, we will document the change below and sign-off showing that the amendment has been approved.

Typed Name	Signature	Date
Brody Looney	Brody Looney Digitally signed by Brody Looney Date: 2023.03.09 10:59:02 -05'00'	3/8/2023
Dylan Haughton	Dylan Haughton Digitally signed by Dylan Haughton Date: 2023.03.09 11:04:55 -05'00'	3/8/2023
Sean Staton	Sean Staton Digitally signed by Sean Staton Date: 2023.03.09 11:06:29 -05'00'	3/8/2023
Johnathan Driscoll	Johnathan Driscoll Digitally signed by Johnathan Driscoll Date: 2023.03.09 11:07:46 -05'00'	3/8/2023
Andrew Howe	Andrew Howe Digitally signed by Andrew Howe Date: 2023.03.09 11:03:01 -05'00'	3/8/2023
Benjamin Drozdenko	DROZDENKO.BENJAMIN.MARTIN.1364363709 Digitally signed by DROZDENKO.BENJAMIN.MARTIN.1364363709 Date: 2023.03.09 17:03:56 -05'00'	3/9/2023

Customer Comments:

Appendix B – Team Review Sign-off

Andy Howe, Dylan Haughton, Johnny Driscoll, Sean Staton, and Brody Looney have all reviewed this document. We all have agreed on the content that listed in this document is accurate and complete. We all agree that the format follows the correct structure. The signatures below prove the information stated previously is correct.

Typed Name	Signature	Date
Brody Looney	Brody Looney Digitally signed by Brody Looney Date: 2023.03.09 10:59:23 -05'00'	3/8/2023
Typed Name	Signature	Date
Dylan Haughton	Dylan Haughton Digitally signed by Dylan Haughton Date: 2023.03.09 11:05:16 -05'00'	3/8/2023
Typed Name	Signature	Date
Sean Staton	Sean Staton Digitally signed by Sean Staton Date: 2023.03.09 11:08:08 -05'00'	3/8/2023
Typed Name	Signature	Date
Johnathan Driscoll	Johnathan Driscoll Digitally signed by Johnathan Driscoll Date: 2023.03.09 11:08:28 -05'00'	3/8/2023
Typed Name	Signature	Date
Andrew Howe	Andrew H Digitally signed by Andrew Howe Date: 2023.03.09 11:03:28 -05'00'	3/8/2023
Typed Name	Signature	Date

Comments:

Appendix C – Document Contributions

Group Member	Sections Worked On	Contribution Percentage
Sean Staton	2 Code Inspection Process, 2.1 Description, 2.3 Inspection Meetings	20%
Dylan Haughton	Appendices, 1 Introduction, 4 Defects	20%
Andy Howe	2.2 Impressions of the Process	20%
Johnny Driscoll	3.0 Modules Inspected	20%
Brody Looney	1.1 Purpose of This Document, 1.1a References, 1.2 Coding and Commenting Conventions, 1.3 Defect Checklist	20%