Test Plan

For an Intrusion Detection System using a Neural Network

Version 2.0

Submitted in partial fulfillment of the requirements of the degree of MSE

Blake Knedler

CIS 895 – MSE Project

Kansas State University

Table of Contents

1 Introduction

This document will provide the test plan for the PyIDS – a python interpretation of an intrusion detection system. The intrusion detection system is a single component itself but consists of several pieces that work together to perform the required functionality. This document will detail the testing for each of the components of the system as well as the overall system. The paper will lay out how each of the required shalls will be tested and what the expected outcome should be.

2 Packages Tested

The Intrusion Detection System consists of only three main pieces of functionality. The description of each of these are detailed in the system architecture design. Overall, the system is simple from the system level standpoint due to the fact that there are only three main packages. However, each of these packages are very complex in nature causing the overall view of the system to be very complex. The individual packages being tested are below:

1. Data Traffic Reader
   - Traffic Reader
   - Traffic Interpreter
   - Traffic Storage
2. Neural Network
   - Input Layer
   - Hidden Layer
   - Output Layer
   - Backpropogation Training
3. Recorder
   - Notifier
   - Logger
   - User Interface

3 Tested Requirements

In this section, we will look at the different requirements that will be tested. Only some of the requirements are critical requirements. Testing of non-critical requirements will only happen if those non-critical features are implemented. Some of these may not be implemented due to timing constraints. The term “shall” is binding to the requirement of the application. The terms will and may are not binding to the requirement of the application.

3.1 SR1.1 [Critical Requirement]

- The system shall be able to read data from the host system network card.

3.2 SR2.1 [Critical Requirement]

- The system shall be able to interpret the data from the received network traffic and store it in a usable format.

3.3 SR3.1 [Critical Requirement]

- The system shall be able to determine if the network data received by the host machine is malicious with at least 85% accuracy.

3.4 SR3.2

- The system shall determine what type of attack is being made to the host network when malicious network traffic is found.

3.5 SR4.1 [Critical Requirement]

- The system shall be able to train itself through backpropagation on known network traffic data.

3.6 SR5.1 [Critical Requirement]

- The system shall be able to notify the User of the host system when a malicious attack is encountered.

3.7 SR6.1

- The system shall be able to log all malicious attacks into a log file.

4 Approach

For this application, only functional unit testing and system testing will be performed. The unit testing testing will be performed on each individual component and the system testing on the system as a whole.

The unit tests will be done by using a python unit test framework unittest. This unit test framework will allow for a white box type testing allowing access to the inner workings of the code and allowing for more accurate testing. The tests will test the requirements that are critical and that are known to be implemented. Each of the unit tests may test multiple test scenarios since some scenarios partially rely on initialized parts of the system to be as accurate as possible.

5 Pass/Fail Criteria

Each test will be rated on either pass or fail. These tests will aim at proving the pass/fail of each of the specified requirements in this document and in the Vision Document. Each test will document which requirement(s) it is trying to cover. If the test does not meet all the requirements it is testing for then the system fails that test completely. In order to aid in this testing effort many of the tests will only test a single requirement to help with determining the root cause of a failure if it were to happen.

6 Test Scenarios

This section will layout each of the different testing scenarios.

1. Test Case 1: Read Data Array

- Requirement(s): SR4.1
- This test will run the Data Reader portion of the system and ensure that the system can accurately read in the data used in the backpropagation aspect of the system.

2. Test Case 2: Read Full Data Array

- Requirement(s): SR4.1
- This test will run the Data Reader portion of the system and ensure that the system can accurately read in the full data used in the backpropagation aspect of the system.

3. Test Case 3: Read Data

- Requirement(s): SR4.1
- This test will run the Neural Network package standalone. The test will consist of feeding the Neural Network with test data, capturing the decisions made, and determining that correctness by comparing the decisions made to the actual results. It will then check to determine if the correct attack type was reported compared to the expected outcome.

4. Test Case 4: Malicious Packet Detection

- Requirement(s): SR3.1, SR4.1
- This test will run the Neural Network package standalone. The test will consist of training the Neural Network with the saved training data. The Neural Network will then be tested to ensure it can detect a malicious formed packet. If the Neural Network can detect the malicious packet it will pass.

5. Test Case 5: Neural Network Training

- Requirement(s): SR3.1, SR4.1
- This test will run the Neural Network package standalone. The test will consist of training the Neural Network with the saved training data. The Neural Network will then be tested to ensure it can achieve an accuracy rating of 85% or higher on the same set of data. If the Neural Network can achieve at least 85% accuracy on that set of data, it will pass.

6. Test Case 6: Receive Packet

- Requirement(s): SR1.1
- This test will test the Network Traffic Reader ensuring that it can receive a packet off of the network card that is either TCP or UDP.

7. Test Case 7: Get Single Packet with Data

- Requirement(s): SR2.1
- This test will test the Network Traffic Reader ensuring that it can receive a packet off of the network card that is either TCP or UDP and package the data to be useful for the Neural Network portion of the system.

8. Test Case 8: System Test

- Requirement(s): SR5.1, SR6.1
- This test will be a full simulation test initializing and running the application. This test will be mainly aimed at testing the GUI portions of the system. Mainly ensuring that the system can report notifications and write a log out.

7 Test Deliverables

A log of each test will be documented. The log will consist of how each test was run, what requirements were being tested, and the result of the test. If a test fails, an analysis of the failure will be reported in the test log as well. The analysis will state the cause of the failure and the plan to correct any issues.

8 Environment Needs

The only requirement needs are those needed to run the application itself. There will be no additional needs to run the tests other than the software itself. The application is being developed on a Windows 10 computer using Visual Studio Community 2015 IDE. The application is being written in the Python programming language. All of these environment choices are laid out in the Vision Document of this project.