Test Plan

For an Intrusion Detection System using a Neural Network

Version 1.0

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 3](#_Toc464495402)

[2 Packages Tested 3](#_Toc464495403)

[3 Tested Requirements 4](#_Toc464495404)

[3.1 SR1.1 [Critical Requirement] 4](#_Toc464495405)

[3.2 SR2.1 [Critical Requirement] 4](#_Toc464495406)

[3.3 SR3.1 [Critical Requirement] 4](#_Toc464495407)

[3.4 SR3.2 4](#_Toc464495408)

[3.5 SR4.1 [Critical Requirement] 4](#_Toc464495409)

[3.6 SR5.1 [Critical Requirement] 4](#_Toc464495410)

[3.7 SR6.1 4](#_Toc464495411)

[4 Approach 5](#_Toc464495412)

[5 Pass/Fail Criteria 5](#_Toc464495413)

[6 Test Scenarios 6](#_Toc464495414)

[1. Test Case 1: Capturing Data 6](#_Toc464495415)

[2. Test Case 2: Correctness Accuracy 6](#_Toc464495416)

[3. Test Case 3: Attack Type 6](#_Toc464495417)

[4. Test Case 4: Backpropogation Training 6](#_Toc464495418)

[5. Test Case 5: Notification 6](#_Toc464495419)

[6. Test Case 6: Logging 6](#_Toc464495420)

[7. Test Case 7: System Test 7](#_Toc464495421)

[7 Test Deliverables 7](#_Toc464495422)

[8 Environment Needs 7](#_Toc464495423)

# 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.

# 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

# 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.

### SR1.1 [Critical Requirement]

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

### 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.

### 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.

### SR3.2

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

### SR4.1 [Critical Requirement]

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

### SR5.1 [Critical Requirement]

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

### SR6.1

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

# Approach

For this application, only functional black box testing will be performed. The black box testing will be performed on each individual component and on the system as a whole. Black box testing is a method of testing that does not know the internals to the system in test but only cares about the inputs and outputs. Additionally, some extra features will be added to the system to aid in testing.

The overall system black box testing will be done by running the system as normal with a few additional features turned on to prove some of the requirements. Some of these features may include saving all received data and self-injecting malicious data. This will allow me to determine if the system is recognizing the data and determining if it is accurately meeting the correctness goal. Also, it will allow me to insert malicious data since I do not want to actually perform any malicious attacks on myself.

The separated package black box testing will be done by running the packages individually with a temporary main. These tests will ensure that the data coming out of each package is accurate and meeting the specified requirements. Additional features may be added to the temporary main statements to capture all the input and output to each individual package.

# 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.

# Test Scenarios

This section will layout each of the different testing scenarios.

### Test Case 1: Capturing Data

* Requirement(s): SR1.1; SR2.1
* This test will run the Data Traffic Reader and determine if it can capture the data by capturing the output of this package after it reads the data off the network card. The analysis will look at the format of the data to ensure that the data is formatted correctly.

### Test Case 2: Correctness Accuracy

* Requirement(s): SR3.1
* 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.

### Test Case 3: Attack Type

* Requirement(s): SR3.2
* 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.

### Test Case 4: Backpropogation Training

* Requirement(s): 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. If the Neural Network can achieve a correctness similar to Test Case 2 then it will pass the test.

### Test Case 5: Notification

* Requirement(s): SR5.1
* This test will run the Recorder Package standalone. It will test that it creates a notification when the package receives an input notifying it of a malicious data packet. The notification must have some level of detail about the packet.

### Test Case 6: Logging

* Requirement(s): SR6.1
* This test will be similar to Test Case 5 but will check a log file instead of a notification.

### Test Case 7: System Test

* Requirement(s): N/A
* This test will be a full simulation test initializing and running the application. From a separate application I will inject malicious packets to fake malicious data into the system.

# 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.

# 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.