System Architecture Design

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 3](#_Toc469165856)

[2 Architecture 3](#_Toc469165857)

[3 Component Design 4](#_Toc469165858)

[3.1 Component Diagram 4](#_Toc469165859)

[3.2 Component Interface Specification 4](#_Toc469165860)

[4 Package Design 6](#_Toc469165861)

[4.1 System Context Package Diagram 6](#_Toc469165862)

[4.1.1 Network Traffic Reader 6](#_Toc469165863)

[4.1.2 Neural Network 7](#_Toc469165864)

[4.1.3 Backpropogation Trainer 7](#_Toc469165865)

[4.1.4 Recorder 7](#_Toc469165866)

[5 Class Design 9](#_Toc469165867)

[5.1 Class Design Diagram 9](#_Toc469165868)

[5.1.1 Network Traffic Reader 10](#_Toc469165869)

[5.1.2 Recorder 11](#_Toc469165870)

[5.1.3 Neural Network 15](#_Toc469165871)

[5.1.4 Data Reader 16](#_Toc469165872)

[5.1.5 PyIDS (UI) 18](#_Toc469165873)

[6 Sequence Design 22](#_Toc469165874)

[6.1 Operating Sequence Diagram 22](#_Toc469165875)

[6.2 Train Sequence Diagram 23](#_Toc469165876)

[7 Formal Requirements Specification Design 24](#_Toc469165877)

# Introduction

This document will provide the system design information 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 detailed design of the system using component and interface specifications in the form of the standard UML design language.

# Architecture

The Intrusion Detection System architecture is a very simple design. The architecture is a layered approach that is event driven. There are three main layers contained within the IDS. The three layers are the Network Traffic Reader, Neural Network, and Recorder. The Network Traffic Reader is a data reading layer of the system. It will take the data from the network card and package it in a way that is useful to the rest of the system. The Neural Network layer of the system is the brains of the system. It will take the data that is read in the Network Traffic Reader layer and make a decision based on backpropogation training or loaded synapse weights. The final layer is the Recorder. The Neural Network layer will communicate to this layer indicating any malicious packets it has received. It is the Recorder layer’s responsibility to log that information and notify the user. Since the system architecture is simple in nature, the Recorder also acts as the user interface. It will respond to the user when a start or train sequence is requested and notify the other layers of this information.

# Component Design

In this section, we will look at the different components of the system and focus on how they interact with each other through interfaces. We will also look at what interfaces the overall system has with external devices and users.

## Component Diagram

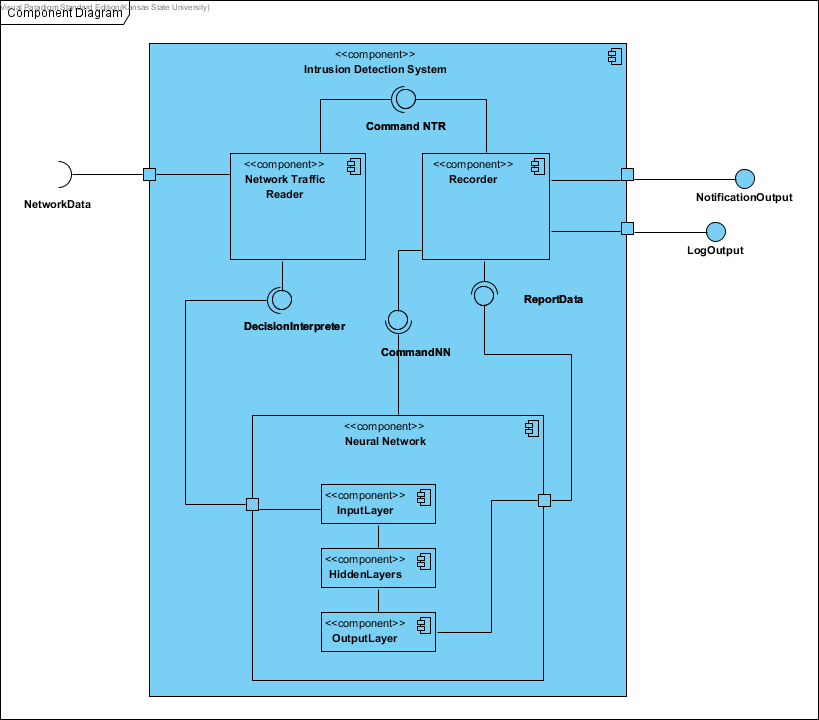


Figure 1. IDS System Component Design

## Component Interface Specification

Figure 1 shows the component diagram of the Intrusion Detection System. There are three main components contained in the overall component of the Intrusion Detection System. There are also three external interfaces to this system which are the NetworkData, which is the data traffic, the NotificationOutput and LogOutput which are notifications to the user about the data the system read. The Network Traffic Reader component is the component of the Intrusion Detection System that will read the network traffic. It will then pass this data via the DecisionInterpreter connection to the Neural Network component. This Neural Network component consists of three sub-components. These components are each of the layers of the Neural Network component. The InputLayer component will receive the data for the Neural Network component and pass the data on to the HiddenLayer and then to the OutputLayer. The OutputLayer then provides the connection of ReportData to the Recorder component. This connection will be how the Neural Network component passes any decisions of malicious data traffic to the recorder. Any non-malicious traffic can also be passed via this interface as well. The final component of this Intrusion Detection System is the Recorder component. This component is responsible for both logging and notifying the user of the data that it received.

# Package Design

In this section, we will go into more detail about each of the packages of the system breaking them down by the different pieces and describing what they are doing for the system and how they are playing a role in the system.

## System Context Package Diagram

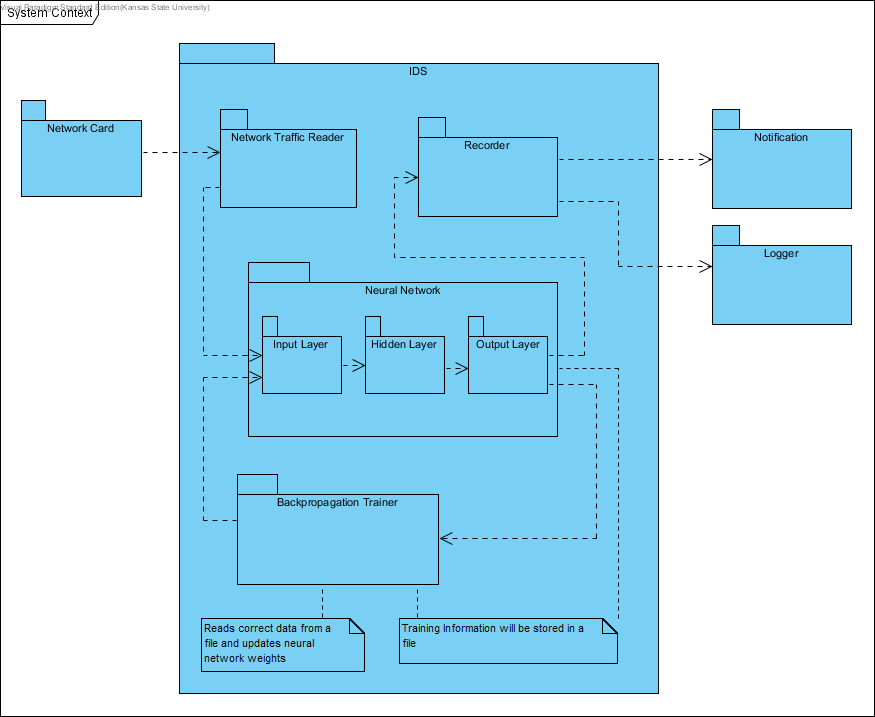


Figure 2. IDS System Package Diagram

### Network Traffic Reader

The Network Traffic Reader package is the data reader for the system. The main goal of this package is to read the network traffic off of the network card as quickly as possible. There are no strict time limits set for this since the configuration of the host system can vary. The Network Traffic Reader package will also be responsible for stripping the network packages as they read to obtain the important information useful to the system. This data will be modeled after the training data to allow for the system to make correct decisions on the packets. This package is very simple in nature and really on performs two real functions. To grab the data and get it into a useable format.

### Neural Network

The Neural Network package is the brains of the system. The Neural Network package reads the data it receives from the Network Traffic Reader package and makes a decision based on its training data to determine if the data packet is malicious or not. The Neural Network consists of three smaller packages. These packages are the individual layers of the Neural Network that work together to appropriately manipulate the data to come to a conclusion for an answer. The Input Layer package will take the data in and store it in a way that is useful to the Neural Network. The Hidden Layer will receive the data from the Input Layer and pass it to the Output Layer for the decision. In the passing of data from one layer to the next a set of weights are applied and functions are used to manipulate the data to find patterns associated with the trained data. This helps determine what the outcome of the data should be. The Output Layer will receive a single answer about each packet. This answer will be if the data packet is malicious or not. The Output Layer is also in charge of letting the Recorder package know if a malicious packet has been detected. The overall goal of the Neural Network is to make a decision on the data packet coming in and letting the Recorder package know if it has found a malicious packet.

### Backpropogation Trainer

The Backpropogation Trainer package is in charge of training the Neural Network package. This is an extremely crucial part of the system that is enforced prior to running the system to capture live data. The system will have a file containing the training data which will be used by this package. The package will feed the Neural Network package the data and assess the correctness of the Neural Network. The package will continue to train the Neural Network package until it has reach a certain level of validity with the answers. The Backpropogation Trainer will also allow the Neural Network to store a file of previously trained weights. This allows the system to only need to be trained once and to reuse that training information for future sessions.

### Recorder

The Recorder package is the user interface portion of the system. This package is in charge of reporting any malicious packets found by the system and logging that information. The Recorder will also handle the other small user interface connections that are needed to start, stop, and train the system via a user interface. The main task of the Recorder package is to take the data from the Neural Network and inform the user of the malicious data that was found. This data will also be stored in a log file for future reference to the user. Since there is not a user interface packet due to the small amount of user interface needs, the Recorder will act as the user interface. This package will be in charge of taking the user input to start the system, stop the system, train, etc. It will then relay that information to the rest of the system to begin the task that was requested.

# Class Design

In this section, we will look briefly at a class design diagram of the three components of the system. Since this is a high level architecture document, this section will not completely detail each of these classes.

## Class Design Diagram

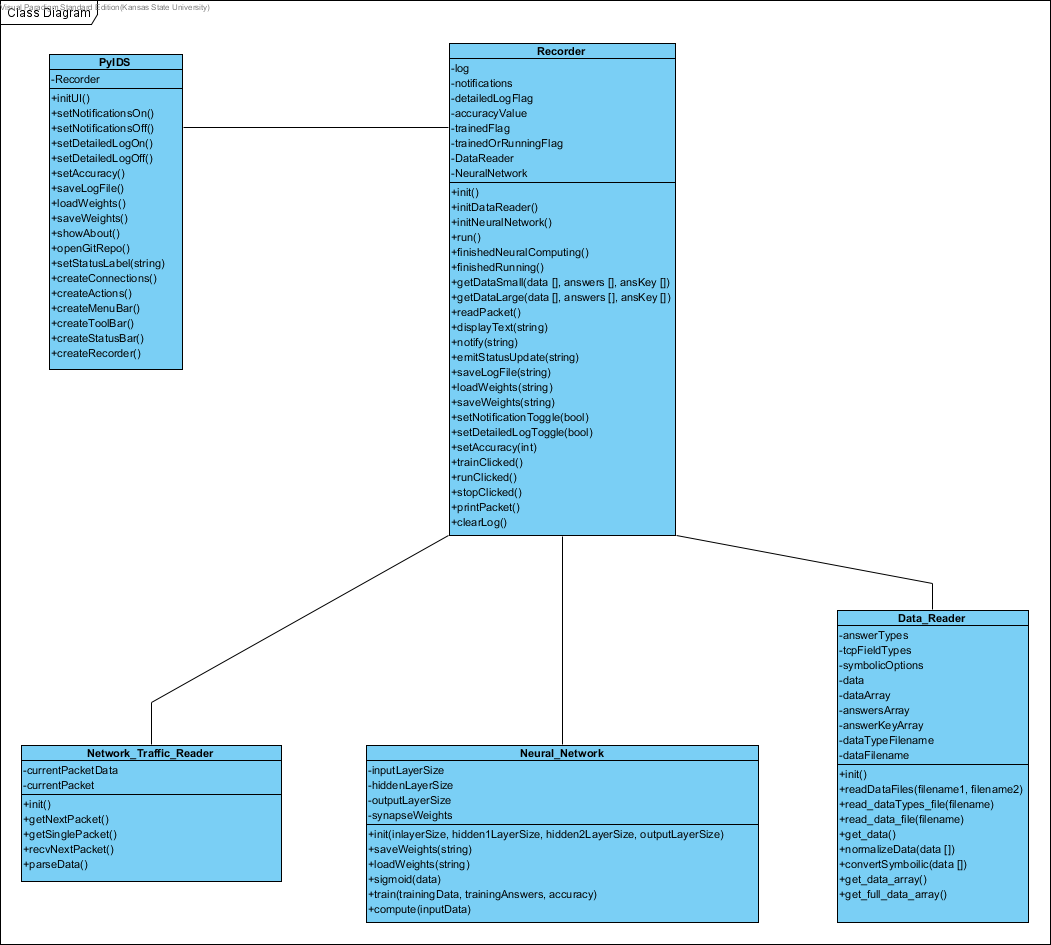


Figure 3. IDS System Class Diagrams

Figure 3 shows a class diagram of the three main components of the system. As mentioned previously, these three components are the Recorder, the Neural Network, and the Network Traffic Reader.

### Network Traffic Reader











### Recorder















































### Neural Network













### Data Reader



















### PyIDS (UI)





































# Sequence Design

In this section, we will look at a couple of the main operating sequences and how system communicates between the different internal components and also to any external user or device.

## Operating Sequence Diagram

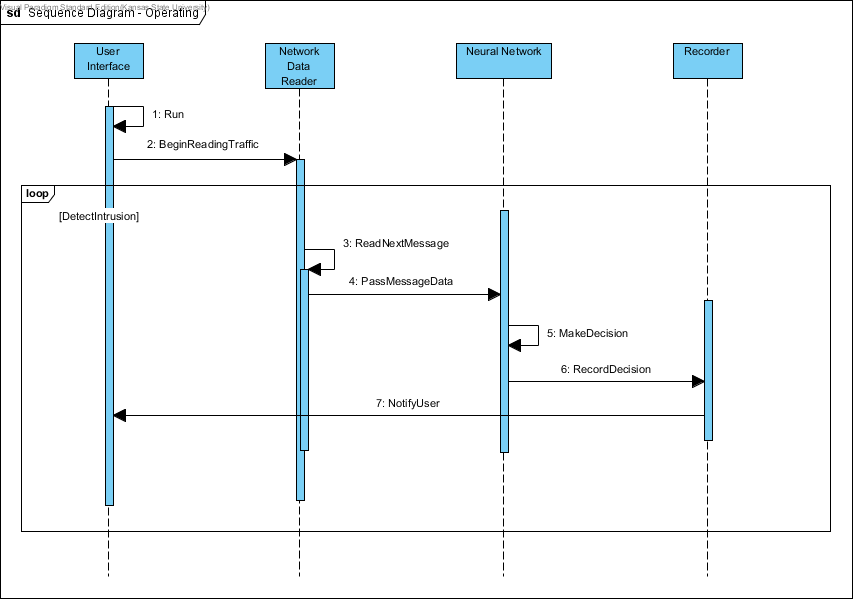


Figure 4. IDS System Operating Sequence Diagram

Figure 4 shows a sequence diagram of the main operating sequence for the Intrusion Detection System. The sequence begins with the user selecting to begin running the system. The user interface is separated in the diagram to help distinguish between it and the Recorder functionality. As mentioned previously, these two aspects will be handled by the Recorder. The User Interface notifies the Network Data Reader to begin reading network traffic. As each message is read, the Network Data Reader will pass the message data to the Neural Network. The Neural Network will then make a decision about the data packet and notify the Recorder of malicious packets. The Recorder will then notify the user by a notification system and a logging system. This process is an iterative process for each message read by the Network Data Reader.

## Train Sequence Diagram

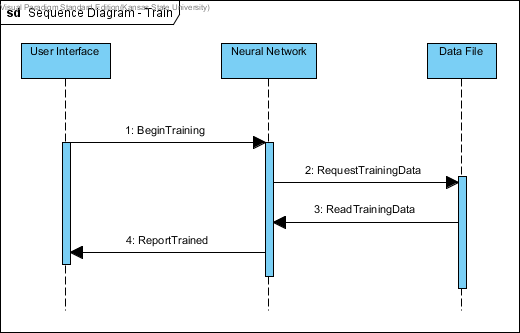


Figure 5. IDS System Train Sequence Diagram

Figure 5 shows a sequence diagram of the training scenario. This sequence begins with the user selecting to begin training. The User Interface notifies the Neural Network to begin training. The Neural Network will read the Data File containing the training data. It will then evaluate its performance of the training information and notify the user that it has now been trained and is ready to start.

# Formal Requirements Specification Design

--

-- PyIDS Formal Specification

--

-- pyIDS\_fs.use

--

-- This is a formal specification for the

-- Python Intrusion Detection System (PyIDS).

--

--

-- Author: Blake Knedler

-- Date: September 11, 2016

--

model PyIDS

-- ===========================================================

-- E N U M E R A T I O N S

-- ===========================================================

-- Enumeration list of packet types

enum PacketType {Valid, Malicious}

--

-- C L A S S E S

--

-- Packet class

-- - basic definition of what a packet will be

class Packet

attributes

data : String

type : PacketType

end

-- PacketReader class

-- - will read the packets as they are received

class PacketReader

attributes

currentPacket : Packet

operations

readPacket(p : Packet) : Boolean =

self.currentPacket = p

end

-- NeuralNetwork class

-- - will make decisions based on the validity of the packet

class NeuralNetwork

attributes

decision : Boolean

operations

-- Decision will be true if the packet is malicious

makeDecision(p : Packet) : Boolean =

if (p.type = #Malicious) then

true

else

false

endif

end

-- Notifier class

-- - will notify user of the malicious packets

class Notifier

attributes

notification : String

numberOfMaliciousPackets : Integer

end

--

-- A S S O C I A T I O N S

--

-- Association relating the reader (PacketReader)

-- to reading Packets

association ReadPackets between

PacketReader [1] role reader

Packet [\*] role packet

end

-- Association relating the reader (PacketReader)

-- and decision maker (NeuralNetwork)

association DetermineDecision between

PacketReader [1] role reader

NeuralNetwork [1] role decider

end

-- Association relating the decision maker (NeuralNetwork)

-- to the notificaiton system (Notifier)

association Notifications between

NeuralNetwork [1] role decider

Notifier [1] role notifier

end

--

-- C O N S T R A I N T S

--

constraints

-- There is only one Packet Reader

context PacketReader

inv Only\_One\_PacketReader:

PacketReader.allInstances->size = 1

-- There is only one Neural Network

context NeuralNetwork

inv Only\_One\_NeuralNetwork:

NeuralNetwork.allInstances->size = 1

-- There is only one Notification system

context Notifier

inv Only\_One\_Notifier:

Notifier.allInstances->size = 1

-- Invariant to report a malicious packet

context p:Packet

inv ReportMalicousPacket:

(NeuralNetwork.allInstances)->forAll(nn |

( if nn.decision = true then

nn.notifier.notification = p.data and

nn.notifier.numberOfMaliciousPackets = nn.notifier.numberOfMaliciousPackets + 1

else

nn.notifier.numberOfMaliciousPackets = nn.notifier.numberOfMaliciousPackets

endif ))