Chapter 5

IMPLEMENTATION

This chapter contains the language details and the libraries used for the implementation. Additionally, the pseudo code of the implementation has been outlined in the following sections.

5.1 Language used for implementation

The core system of the project has been written in Python, whereas the front end contains a mix of Python, HTML and CSS.

5.1.1 Python

Python[22] is a widely used general-purpose, high-level programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C. The language provides constructs intended to enable clear programs on both a small and large scale.

Python supports object-oriented, imperative and functional programming. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library. Like other dynamic languages, Python is often used as a scripting language, but is also used in a wide range of non-scripting contexts. Using third-party tools, such as Py2exe or Pyinstaller, Python code can be packaged into standalone executable programs. Python interpreters are available for many operating systems.

Python is a multi-paradigm programming language: object-oriented programming and structured programming are fully supported, and there are a number of language features which support functional programming and aspect-oriented programming (including by meta-programming and by magic methods). Many other paradigms are supported using extensions, including design by contract and logic programming.

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. An important feature of Python is dynamic name resolution (late binding), which binds method and variable names during program execution. The core philosophy of the language is summarized by the document

"PEP 20 (The Zen of Python)", which includes aphorisms such as:

1. Beautiful is better than ugly
2. Explicit is better than implicit
3. Simple is better than complex
4. Complex is better than complicated
5. Readability counts

Libraries like NumPy, SciPy and Matplotlib allow Python to be used effectively in scientific computing, with specialized libraries such as BioPython and Astropy providing domain-specific functionality.

5.2 Libraries used for implementation

The project uses the following libraries, which extend the functionality of Python's standard library.

5.2.1 NumPy

Arrays in Python do not have the same definition as that of in traditional programming languages like C. In C arrays are used to store data having the same data type. Python arrays also called lists are similar to C arrays except that they can be used to store mixed type of data. NumPy is an extension to the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large library of high-level mathematical functions to operate on these arrays.

NumPy[23] targets the CPython reference implementation of Python, which is a non-optimizing bytecode compiler/interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. NumPy seeks to address this problem by providing multidimensional arrays and functions and operators that operate efficiently on arrays. Thus any algorithm that can be expressed primarily as operations on arrays and matrices can run almost as quickly as the equivalent C code.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars. In comparison, MATLAB boasts a large number of additional toolboxes, notably Simulink whereas NumPy is intrinsically

integrated with Python, a more modern, complete, and open source programming language.

5.2.2 Flask

Flask[24] is a lightweight web application framework written in Python and based on the Werkzeug WSGI toolkit and Jinja2 template engine. It is BSD licensed. Flask takes the flexible Python programming language and provides a simple template for web development. Once imported into Python, Flask can be used to save time building web application.

Flask is called a micro-framework because it keeps the core simple but extensible. There is no database abstraction layer, form validation, or any other components where third-party libraries already exist to provide common functionality. However, Flask supports extensions, which can add such functionality into an application.

Flask has the following features:

1. Contains development server and debugger
2. Integrated support for unit testing
3. RESTful request dispatching
4. Uses Jinja2 for templating
5. Support for secure cookies (client side sessions)
6. 100% WSGI 1.0 compliant
7. Unicode-based
8. Extensive documentation
9. Extensions available to enhance features desired

In the project Flask's server is used to host the front-end. It listens to HTTP requests and responds to them. Based on the request it calls the various modules of the Naive Bayes Classifier and displays the results as a web page via the HTTP response.

5.2.3 Matplotlib

Matplotlib[25] is a plotting library for the Python programming language and its NumPy numerical mathematics extension. It provides an object-oriented API for embedding plots into applications. The pylab interface makes matplotlib easy to learn for experienced MATLAB users, making it a viable alternative to MATLAB as a teaching tool for numerical mathematics and signal processing.

Some of the advantages of the combination of Python, NumPy and matplotlib over MATLAB include:

1. Based on Python, a full-featured modern object-oriented programming language suitable for large-scale software development
2. Free and open source
3. Native SVG support

5.3 Pseudo code

The pseudo code for various algorithms are mentioned in this section.

5.3.1 Determine optimal bands procedure

This module groups the events into bands. The band formation is based on the Jump clustering algorithm.

```
Function determineOptimalBands(S.W[], K)
Input: Array A = {a0..aN}, window size for stream N, number of bands K
Output: 2-dimensional array band, containing indices of per group elements in A
    1. J[N] ← {0} //store jumps between consecutive values of A
    2. Js[K − 1] ← {0} // stores the top (K − 1) jumps
    3. upper, lower, nelements ← 0
    4. I[K − 1] ← 0, Is[K − 1] ← 0
       // stores indices in A of top ((K − 1)) jumps.
    5. Sort A in descending order and copy into array As
      // Determining jumps between consecutive values of As
    6. for i = 0 to N do
    7.   J[i] ← As[i] − As[i + 1]
    8. end
    9. Determine the top K − 1 jumps from array J
    10. Store them in Js[0], Js[1] . . . Js[K − 2]
    11. Store indices of Js[0] to Js[K − 2] in J into array I
    12. Sort array I in ascending order and store in Is
    13. for i = 0 to K − 1 do
```

```
14.   upper ← Is[i]
15.   nelements ← upper - lower + 1
16.   k ← 0
17.   band[i][k++] ← nelements
18.   for j = lower to (lower+nelements) do
19.         band[i][k++] ← index of As[j] in A
20.   end
21.   lower ← upper + 1
22. end
23. band[i][K++] ← remaining elements of A
24. return band
```

5.3.2 Update probabilities procedure

This module computes and updates the probability based on the events.

Function updateProbabilities(S, K, band, L)

Input: TCP Stream structure S, 2-D array band containing band indices of groups of A, number of bands K, Index of observable type L (for writing into appropriate index at S.W)

Output: Probabilities of events updated into corresponding W index in S

```
1. avgwcount ← 0;
  /* Increment total window to accommodate smoothing */
2. S.total windows in learning+= K;
3. for j = 0 to K do
4.   nelband ← band[j][0];
5.   for k = 1 to (1+ nelband) do
6.         avgwcount ← avgwcount + S.W[L][band[j][k]];
7.   end
     /* Smooth the event group and compute mean */
8.   avgwcount++;
9.   avgwcount ← avgwcount / nelband
     /* Write the mean value into the corresponding W indices of the
     observable type. */
10.    for k = 1 to (1+ nelband) do
```

```
11.         S.W[L][band[j][k]] ← avgwcount;
12.     end
      /* Compute probabilities */
13.   for k = 1 to (1+ nelband) do
14.         S.W[L][band[j][k]] ← S.W [L][band[j][k]] /
                                    (S.total windows in learning)
15.   end
16.   avgwcount ← 0
17. end
18. return S
```

5.3.3 Train procedure

This module trains the Naive Bayes Classifier.

Function Train(S, TF, N, K)

Input: Stream S, Traffic Statistics TF, Window size for stream S.N, number of bands S.K

Output: Updated Stream Data Structure S with learnt probabilities.

```
1. for Every window in TF do
2.    Populate S.Cis for each of the 6 Ti s
3.    for i = 1 to 6 do
      /* Update corresponding event occurrence. */
4.          Increment S.W[i][S.Ci]
5.    end
6.    Increment S.total windows in learning
7.    Reset S.Cis
8. end
/* Determine Optimal bands for each row of W, compute
probabilities */
9. for i = 1 to 6 do
10.   band ← determineOptimalBands (S.W[i],S.N,S.K)
      // Determine band ranges for row i
11.   S ← updateProbabilities(S, S.K, band, i)
```

```
            // Determine probabilities for row i
      12. end
      13. return S
```

5.3.4 Train phase procedure

This module initiates training. It calls the training module that trains the NBC against the training dataset. It also calls the module that determines the threshold probability.

```
Function Trainphase()
Input: Partially update stream structure S, Traffic statistics TF, window
size S.N, number of bands S.K, error proportion S.t
Output: Updated Stream Data Structure S with learnt probabilities and
threshold probability for stream
      1. S ← Train(S, TF, S.N, S.K)
        /* Determine Threshold probability for the stream */
      2. S.thresholdprobability ←
            determineThresholdProbability(S.N,TF,S.t)
      3. return S
```

5.3.5 Determine probability procedure

Probability computation involves computing statistics of a window and determining the probability with which the window would have occurred in the training phase.

```
Function determineProbability(W, S)
Input: Stream with NB probabilities S, packet window statistics W
Output: Probability P of window W
      1. Initialize P to 1
      2. Determine counts of packets with flags of all 6 groups  T1 to T6
         and update counts C1 to C6
      3. for i = 1 to 6 do
      4.    P = P * S.W[i][Ci]
      5. end
6. return P
```

5.3.6 Determine threshold probability procedure

This module computes the threshold probability which will be used to classify if a particular window is either an attack window or not.

Function determineThresholdProbability(S, N, TF, t)

Input: Stream S, Window size for stream N, Traffic statistics TF, Error proportion t percent

Output: Threshold probability P_t for S

```
1. Seed Rng ()
2. parray[] ← 0
   // Grouping Learning traffic
3. Split traffic TF uniformly at random into 10 groups G1 to G10
4. for i = 1 to 10 do
5.     G ← TF − Gi
6.     B ← Gi
       /* Learn from 9/10th of traffic
7.     Stemp ← Train (SF, G)
       // Compute probability of remaining 1/10th of traffic
8.  for every window W in B do
9.     P ← determineProbability (W, Stemp)
10.   add P to parray;
11.  end
12. end
   //Sieving lower probabilities
13. result ← (t/100 * len(parray) + 1)th smallest element of parray
14. return result
```

5.3.7 Deploy procedure

This module tests the input traffic and computes if a window is clean or not.

Function Deploy(S, IT, AWC)

Input: Stream S, Input Traffic IT, Abnormal Window Count (AWC)

```
1. A ← 0;
2. for every packet window W in T do
```

```
3.     P ← determineProbability (W, S)
4.    if P < Threshold probability of S then
5.          increment A
6.     else
7.          decrement A
8.     end
9.     if A ≥ AWC then
10.         return attack
11.    end
12. end
```