Chapter 3

SYSTEM ANALYSIS

The hardware and software details, functional requirements and the input to the proposed system is outlined in this chapter. The input details consist of the dataset statistics and the parsing techniques used.

3.1 Requirements

The different hardware and software requirements are covered in the next sub-sections. In the subsequent section, we cover the various functional requirements.

3.1.1 Hardware Requirements

- i386 or x64 based Processor
- 50 GB Secondary storage
- 8 GB of RAM

3.1.2 Software Requirements

- GNU/ Linux based OS
- Python 2.6 or greater
- Pycharm community edition
- Numpy 1.7 or greater
- Wireshark
- Git
- Flask
- Matplotlib
- Firefox or Chrome browser

3.1.3 Functional Requirements

Common functional requirements of an Intrusion Detection System are discussed below.

a) The IDS must continuously monitor and report intrusions.

b) The IDS should be modular and configurable as each host and network segment will

require their own tests and these tests will need to be continuously upgraded and eventually replaced with new tests.

c) The IDS must operate in a hostile computing environment and exhibit a high degree of fault-tolerance and allow for graceful degradation.

d) The IDS should be adaptive to network topology and configuration changes as computing elements are dynamically added and removed from the network.

e) Anomaly detection systems should have a very low false alarm rate.

f) The IDS should be capable of providing an automated response to suspicious activity.

g) The ability to detect and react to distributed and coordinated attacks.

h) The IDS should be able to work with other Commercial Off-the-Shelf (COTS) security tools, as no vendor toolset is likely to excel in or to provide complete coverage of the detection, diagnosis, and response responsibilities.

i) IDS data often requires additional analysis to assess any damage to the network after an intrusion has been detected.

3.2 Dataset

The IDS is trained and tested with standard datasets like DARPA[19], KDD[20] and Auckland[21]. Each dataset is a collection of network packets grouped to form **tcpdump** files. Of the different types of packets in the tcpdump files, we filter only **TCP** packets. The structure of a TCP packet is shown in figure 3.1.

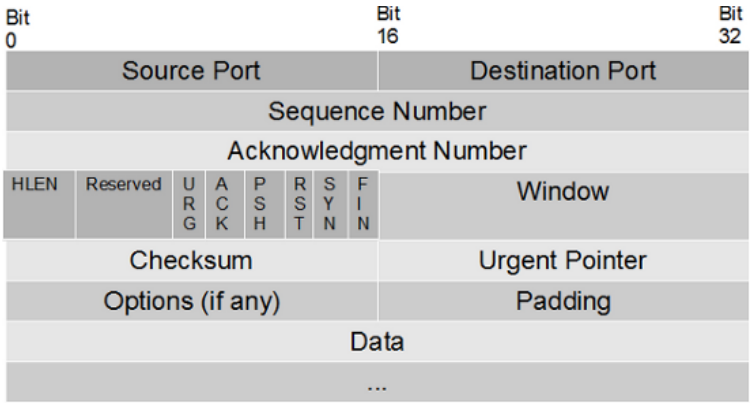

Figure 3.1 TCP Packet Format

Each TCP packet consists of TCP flags like URG, ACK, PSH, RST, SYN and FIN. We filter the flags and group them into six categories as follows in table 3.1:

Type	Flag Set	Flag Name	Description
T1	RST	Reset flag	This flag is set when the remote host rejects the incoming packet from a client on account of trying to establish a connection. This indicates that the remote host has **reset** the connection
T2	SYN	Synchronization flag	This flag is set when the host is trying to establish a connection by sending **SYN** (synchronize) packets
T3	ACK	Acknowledgement flag	This flag is set to **acknowledge** the successful receipt of packets
T4	FIN/ACK	Finish & Acknowledgement flag	This flag is set when the destination host wants to close the connection with the source host after data transfer. The FIN flag is set to indicate **termination** of connection while the ACK flag is set to acknowledge the last packet received during data transfer
T5	PSH/ACK	Push & Acknowledgement flag	This flag is set when feedback is required immediately from the client and server instead of waiting for the buffer to become full before transmission. The data is **immediately pushed** when the PSH flag is set and the ACK flag is set to acknowledge the data received
T6	Others	Others	Any **other flag/s** set other than the above mentioned flags, fall under this category

Table 3.1 TCP Flag Groups

The flags are stored as hexadecimal values which are passed to the program for classifying the incoming traffic as legitimate or malicious. In order to extract only the TCP flags from the datasets we run the following commands -

- ***tcpdump -r inside/w1mon.tcpdump -w test.pcap***

  This command is used to read the data present in a tcpdump file (w1mon.tcpdump) and write the contents to a pcap file (test.pcap)

- ***tshark -r test.pcap -Y tcp -T fields -E separator=/s -e ip.dst -e tcp.dstport -e tcp.flags > op_mon***

  This command takes test.pcap file as input, filters the tcp packets, extracts destination IP, destination port and TCP flags and stores the output in op_mon

- ***cat op_mon | grep "172.16.112.50 23" | cut -d " " -f 3 > op_flag_mon***

  This command selects the particular IP and cuts the third field which contains TCP flag set and stores the output in op_flag_mon

- ***rm op_mon***

  This command removes the intermediate file created

Finally, we get the TCP flags in the form of Hexadecimal values which are used to train and test the IDS.

The following table 3.2 consists of information about the number of attack windows and normal windows present in the dataset files considered to train and test the IDS.

	Normal Windows	**Attack Windows**
darpaTrain	219600	0
darpaNoAttack	61400	0
darpaAttack	37100	37100
kddTrain	216900	0
kddNoAttack	126400	0
kddAttack	58200	58200
aucklandTrain	219600	0
aucklandNoAttack	126400	0
aucklandAttack	393	2048

Table 3.2 Window statistics of the dataset files