CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, *"A hybrid and enhanced Framework for Visualization and Analysis of Network Traffic Data"*, in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Software Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Dr. Maninder Singh and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Nitin Prakash

(Nitin Prakash Verma)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

[illegible]

(Dr. Maninder Singh)
Associate Professor
Head of Department
Computer Science and Engineering Department

Countersigned by

[illegible]

(Dr. Maninder Singh)
Head 31/6/11
Computer Science and Engineering Department
Thapar University
Patiala

[illegible]

(Dr. S. K. Mohapatra)
Dean (Academic Affairs)
Thapar University
Patiala

A hybrid and enhanced Framework for Visualization and Analysis of Network Traffic Data

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Software Engineering

Submitted By
Nitin Prakash Verma
(800931014)

Under the supervision of:
Dr. Maninder Singh
Associate Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

June 2011

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, *"A hybrid and enhanced Framework for Visualization and Analysis of Network Traffic Data",* in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Software Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Dr. Maninder Singh and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

(Nitin Prakash Verma)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Dr. Maninder Singh)
Associate Professor
Head of Department
Computer Science and Engineering Department

Countersigned by

(Dr. Maninder Singh)
Head
Computer Science and Engineering Department
Thapar University
Patiala

(Dr. S. K. Mohapatra)
Dean (Academic Affairs)
Thapar University
Patiala

ACKNOWLEDGEMENT

No volume of words is enough to express my gratitude towards my thesis supervisor **Dr. Maninder Singh**, Head of Department, Computer Science & Engineering, whose guidance, wisdom and invaluable help has aided me in the completion of thesis. He has helped me to explore numerous topics related to the thesis in an organized and methodical manner and provided me with many valuable insights into various technologies.

I am also thankful to **Mr. Karun Verma**, P.G. Coordinator, for the motivation and inspiration during the thesis work.

I would also like to thank the staff members and my colleagues who were always there at the need of the hour and provided with all the help and facilities, which I required, for the completion of my thesis work.

Most importantly, I would like to thank my parents and the Almighty for showing me the way and encouraging me through the difficult times I encountered during the completion of my thesis work.

ABSTRACT

Computer networks are indispensable part of any organizations now. Same time security threats that can cause massive harm in the network are also increasing. To analyze the activities happening in the network, network logs records are generated .

These network traffic logs are growing large day by day. A capable network produces plethora of data varying from megabytes to gigabytes .At the enterprise level where firewalls ,intrusion detection systems ,antivirus generates millions of alerts daily, situation is more aggravated. It is very cumbersome and unwieldy for network administrator to analyze these voluminous logs and to take security decisions. Network visualization can help administrator to minimize overhead of understanding and analyzing such high dimensional data.

There is much of work going on various network visualization tools, but these tools seldom provide a mechanism to discern what is happening in the network because they present large, redundant or inappropriate data for visualization of the network

In this work, we propose a framework of visualization for a large and diverse set of data logs, based on filtering and reducing the data to discover the concerned data and reduces cognitive burden of the administrator. Further, the framework is implemented and demonstrated over the large network of university.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

This chapter gives a detailed description of Network Visualization and its related aspects. It also describes visualization with respect to security perspective. Here, need of visualization, data sources needed for security data, process of information visualization, are also described.

1.1 Background

Visualization of data enables to communicate and analyze large amount of information. Too often, information is encoded in text. It is more difficult to immediately grasp the essence of something if it is just described in words. In fact, it is hard for the brain to process text. Pictures or images, on the other hand, can be processed extremely well. They can encode a wealth of information and are therefore, well suited to communicate much larger amounts of data to a human. Pictures can use shape, color ,size, relative positioning, and so on to encode information, contributing to increased bandwidth between the information and the consumer or viewer [1].

Visual Analytics is defined in [2] as “the science of analytical reasoning facilitated by active visual interfaces.” It is motivated by the need to gain understanding of features, trends and anomalies present in large and complex data collections. The challenges in network traffic analysis are motivated by a combination of rapid growth in the internet combined with the time-critical nature of responding to problems that arise. According to Burrescia [3], traffic volume of production network servicing the U. S. Department of Energy’s research laboratories, has increased by an order of magnitude every 46 months since 1990. This trend is expected to continue into the foreseeable future resulting in multiple gigabytes’ worth of connection records. A year’s worth of such data currently requires on the order of tens of terabytes or storage [4].

Many disciplines are facing an ever-growing amount of data that needs to be analyzed, processed, and communicated so new ways to work with all this data are needed. People who have to look at, browse, or understand the data need ways to display relevant information graphically to assist in understanding the data, analyzing it, and remembering parts of it. Browsing huge amounts of data is crucial for finding information and then exploring details of a result set. Interaction with the visualizations is one of the key elements in this process. It is not just the expedited browsing capabilities that visualization has to offer, but often a visual representation ,in contrast to a textual representation- helps to discover relationships well hidden in the wealth of data. Finding these relationships can be crucial [1].

1.2 Need of Visualization

A visual approach significantly facilitates the task (as compared to using text based tools).Visualization offers a number of benefits over textual analysis of data. People can scan, recognize, and recall images rapidly. In addition, the human brain is an amazing pattern recognition tool, and it can detect changes in size, color, shape, movement, and texture very efficiently.

Information Visualization takes advantage of human perception. While information visualization technically includes input gathered from any of human senses, it depends largely upon human vision ,a high bandwidth parallel process .Using vision it can rapidly locate ,discover, identify , and compare objects; all of these are essential tasks, considering the overwhelming amount of information and raw data available. In comparison while computers excel at performing precisely defined tasks ,they lack the ability to perform many of the tasks that human can ,even considering state-of-the-art artificial intelligence and pattern making systems [5].

Visualization facilitate to create an image for each question about a dataset. Instead of consuming time on textual data and trying to remember all the relationships between individual entries, an image conveys the data in a concise form. One interesting aspect of visual representations is that they also cause the viewer to pose new questions. A human has the capability to look at a visual representation of data and see patterns. Often, these patterns are not anticipated at the time the visual is generated.[1]

Visual displays provide the highest bandwidth channel from the computer to the human. More information is acquired through vision than through all of the other senses combined. The 20 billion or so neurons of the brain devoted to analyzing visual information provide a pattern-finding mechanism that is a fundamental component in much of cognitive activity. Improving cognitive systems often means tightening the loop between a person, computer-based tools, and other individuals. On the one hand, the human visual system, a flexible pattern finder, coupled with an adaptive decision-making mechanism. On the other hand are the computational power and vast information resources of the computer and the World Wide Web. Interactive visualizations are increasingly the interface between the two. Improving these interfaces can substantially improve the performance of the entire system [6].

A visual representation provides new insights into a given dataset. Different graphs and configurations highlight various different properties in the dataset and help identify previously unknown information. If the properties and relationships were known upfront, it would be possible to detect these incidents without visualization. However, they had to be discovered first, and visual tools are best suited to do so. Interactive visualizations enable even richer investigations and help discover hidden properties of a dataset [1].

Instead of wading through thousands of lines of textual log data, it is much more efficient to graph certain properties of the data to see trends and outliers. The time it takes to analyze the log files is drastically cut down. This frees up people’s time and allows them to think about the patterns and relationships found in the data. It also speeds up the detection of and response to new developments. Fewer people are needed to deal with more data.

1.3 Security Visualization

As an increasing number of software applications are web-based or web-connected, security and privacy have become critical issues for everyday computing. Computer systems are constantly being threatened by attackers who want to compromise the privacy of transactions (e.g., steal credit card numbers) and the integrity of data (e.g., return a corrupted file to a client). Therefore, computer security experts are continuously developing methods and associated protocols to defend against a growing number and variety of attacks. The development of security tools is an ongoing process that keeps on reacting to newly discovered vulnerabilities of existing software and newly deployed technologies [7].

1.3.1 Data sources

Visualization cannot happen without data or information. Therefore, before start knowing about graphs and visualization, one should know about data .There is need to create an understanding of the data sources that are available-

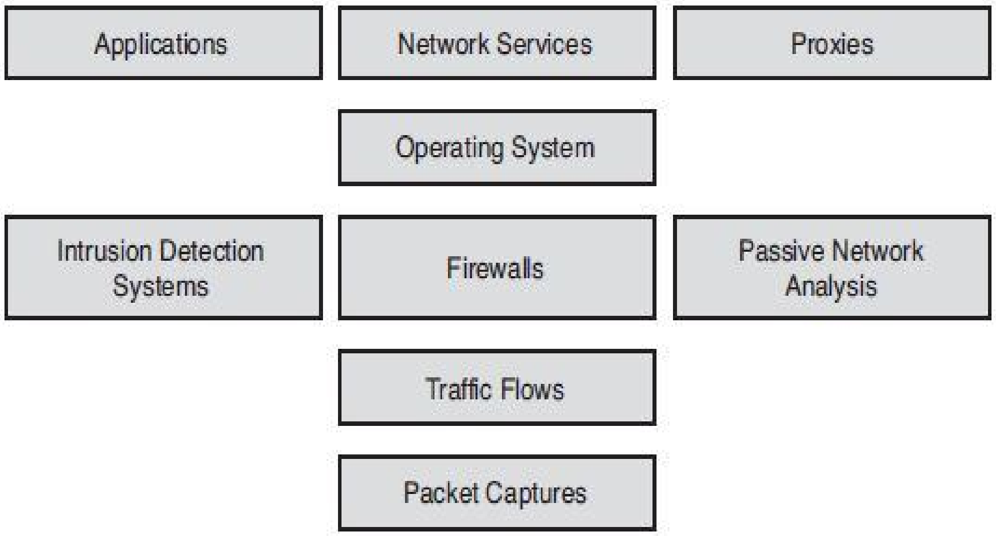


Figure 1.1 : Network stack indicating the data sources [1]

1.3.2 Security Data

Security data is all the data that can help with security analysis, investigations, reporting, or monitoring. Security data is not a distinct class of data. Records from sources such as networking devices (i.e., network flows or routing information), transaction records from financial systems, DHCP logs, and so on are all security data if they help to solve a security problem or answer a security question.

Security data can be separated into two broad categories: time-series and static data. Time-series data is all data that can be attributed to a point in time. Log records fall into this category. Each record generally has a timestamp associated with it, identifying the time the activity was logged .It is not necessarily true that the timestamp identifies the exact time an event (i.e., the physical manifestation of an activity) occurred. In certain cases, the recorded timestamp is the time when the log record was written. Static data is information that has no inherent time associated with it files or documents, for example). Also, any kind of information about the machines in human environment or information about users can be considered static information. In certain cases, static information as time-series data can be utilized. For example, configuration files can be associated with a point in time by utilizing their last modification time [1].

One of the challenges of working with static files is that they span multiple lines. Processing this data is significantly harder than working with, for example, single-line log records. It has to do with parsing of data. Parsing is the process of taking a log record and identifying the individual parts in it. For example, a firewall log entry is given, To generate a statistic over the most active machines, there is need to extract that source address from the entry. This process is called parsing . It is significantly harder to parse a multiline record or a file because the information is not contained in a single line and information is context sensitive for example, in what part of a configuration file a certain statement shows up. The parser therefore has to keep state, which complicates its design. As an example, vulnerability scan recorded as an XML file. If there is requirement to do any analysis of vulnerabilities per host, it need to identify the place in the file where the host information is stored, and then relative to that find the vulnerabilities for the host and stitch that information together. The information is not contained in a single, easy-to-process entry.

For visualization tools to work with data, there is need to convert it to specific formats that the tools understand. Most of them do not contain built-in parsers that can be used to directly read the log files. To use own data with some tools, it must transform log files into these formats. The reason that many tools require different types of inputs is that each tool requires a slightly different set of information to operate.

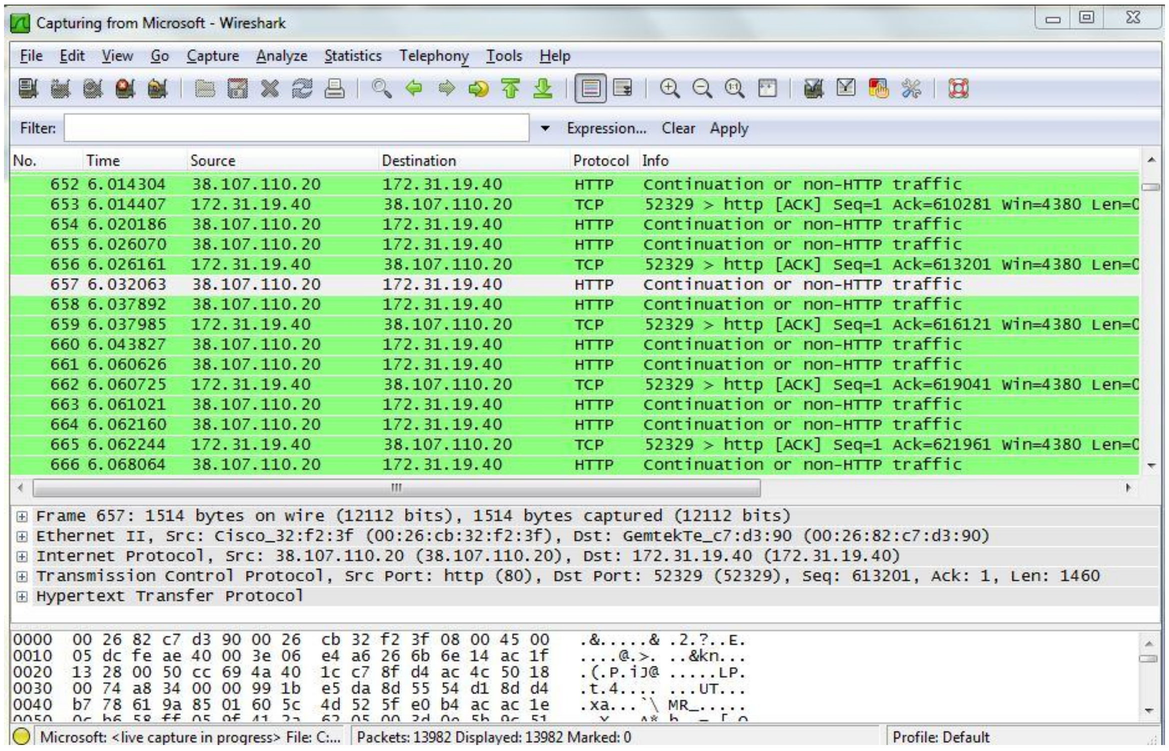
It does not help that there is no currently established common format for writing log records. Parsers are needed for each and every log source to transform the data into the specific format that the visualization tool uses. There have been attempts in the past to standardize log records (for example, the Intrusion Detection Message Exchange Format [IDMEF])[8]. Unfortunately all of those attempts have failed. A new effort was started by MITRE at the beginning of 2007. It is called Common Event Expression (CEE)[9]. The standard is still in its early stages. No specific log formats have been published. Some proposals have been submitted but so far, nothing has been formally accepted. Hopefully, CEE, which is heavily influenced by industry and academic entities, is defining a widely accepted standard and will eliminate the need to write parsers for hundreds of different log formats .Visualization of security data would definitely benefit from such a development.

1.3.3 Packet Captures

In the field of computer network administration, pcap (packet capture) consists of an application programming interface (API) for capturing network traffic. Unix-like systems implement pcap in the libpcap library [10]. Windows uses a port of libpcap known as WinPcap [11]. Monitoring software may use libpcap and/or WinPcap to capture packets travelling over a network and, in newer versions, to transmit packets on a network at the link layer, as well as to get a list of network interfaces for possible use with libpcap or WinPcap.

A network packet is physically received by the network interface. From there, it is passed to the operating system. The network driver in the operating system is responsible for decoding the information and extracting the link-layer headers (e.g., the Ethernet header). From there, the packet is analyzed layer by layer to pass it up in the network stack from one protocol handler to the next. Packet captures are recorded right when the network packet is handed from the network interface to the operating system.

There are many programs that can be used to collect network traffic. The two most common ones are Wireshark[12] and tcpdump[13]. These programs listen on the network interface and display the traffic. Both tools take the raw network traffic and analyze the entire packet to decode the individual network protocols. They then display the individual header options and fields in a more or less human-readable form rather than the original binary format. Wireshark provides a graphical user interface to explore the network packets. It also ships with a command-line tool called tshark. Tcpdump is a command-line-only tool.


Capturing from Microsoft - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter:
Expression... Clear Apply

```
No.   Time      Source          Destination     Protocol Info
  652 6.014304  38.107.110.20   172.31.19.40    HTTP     Continuation or non-HTTP traffic
  653 6.014407  172.31.19.40    38.107.110.20   TCP      52329 > http [ACK] Seq=1 Ack=610281 Win=4380 Len=C
  654 6.020186  38.107.110.20   172.31.19.40    HTTP     Continuation or non-HTTP traffic
  655 6.026070  38.107.110.20   172.31.19.40    HTTP     Continuation or non-HTTP traffic
  656 6.026161  172.31.19.40    38.107.110.20   TCP      52329 > http [ACK] Seq=1 Ack=613201 Win=4380 Len=C
  657 6.032063  38.107.110.20   172.31.19.40    HTTP     Continuation or non-HTTP traffic
  658 6.037892  38.107.110.20   172.31.19.40    HTTP     Continuation or non-HTTP traffic
  659 6.037985  172.31.19.40    38.107.110.20   TCP      52329 > http [ACK] Seq=1 Ack=616121 Win=4380 Len=C
  660 6.043827  38.107.110.20   172.31.19.40    HTTP     Continuation or non-HTTP traffic
  661 6.060626  38.107.110.20   172.31.19.40    HTTP     Continuation or non-HTTP traffic
  662 6.060725  172.31.19.40    38.107.110.20   TCP      52329 > http [ACK] Seq=1 Ack=619041 Win=4380 Len=C
  663 6.061021  38.107.110.20   172.31.19.40    HTTP     Continuation or non-HTTP traffic
  664 6.062160  38.107.110.20   172.31.19.40    HTTP     Continuation or non-HTTP traffic
  665 6.062244  172.31.19.40    38.107.110.20   TCP      52329 > http [ACK] Seq=1 Ack=621961 Win=4380 Len=C
  666 6.068064  38.107.110.20   172.31.19.40    HTTP     Continuation or non-HTTP traffic

⊞ Frame 657: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits)
⊞ Ethernet II, Src: Cisco_32:f2:3f (00:26:cb:32:f2:3f), Dst: GemtekTe_c7:d3:90 (00:26:82:c7:d3:90)
⊞ Internet Protocol, Src: 38.107.110.20 (38.107.110.20), Dst: 172.31.19.40 (172.31.19.40)
⊞ Transmission Control Protocol, Src Port: http (80), Dst Port: 52329 (52329), Seq: 613201, Ack: 1, Len: 1460
⊞ Hypertext Transfer Protocol

0000  00 26 82 c7 d3 90 00 26  cb 32 f2 3f 08 00 45 00   .&.....& .2.?..E.
0010  05 dc fe ae 40 00 3e 06  e4 a6 26 6b 6e 14 ac 1f   ....@.>. ..&kn...
0020  13 28 00 50 cc 69 4a 40  1c c7 8f d4 ac 4c 50 18   .(.P.iJ@ .....LP.
0030  00 74 a8 34 00 00 99 1b  e5 da 8d 55 54 d1 8d d4   .t.4.... ...UT...
0040  b7 78 61 9a 85 01 60 5c  4d 52 5f e0 b4 ac ac 1e   .xa...`\ MR_.....
```

Microsoft: <live capture in progress> File: C:...
Packets: 13982 Displayed: 13982 Marked: 0
Profile: Default

Figure 1.2 : Wireshark packet capture

Commonly, network traffic needs to be recorded for later analysis. The most common format for packet captures is the PCAP format. Most sniffers (or network traffic analysis tools) can read this binary format.

There is need of actual data contained in packet captures that is of interest for visualization and analysis. The following list shows the typical types of information that can extract from packet captures and their meaning:

- Timestamp (**I**) : The time the packet was recorded.
- IP addresses (**II**) : The addresses show the communication endpoints that generated the traffic.
- Ports (**III**) : Network ports help identify what service is used on the network.
- TCP flags (**IV**) : The flags can be used to verify what stage a connection is in. Often, looking at the combination of flags can identify simple attacks on the transport layer.
- Ethernet addresses (**V**) : Ethernet addresses reveal the setup of the local network.
- Packet size (**VI**) : Packet size indicates the total size of the packet that was transmitted.

A sample packet capture, recorded with tcpdump, looks like this:

```
(I)18:57:35.926420 (V) 00:0f:1f:57:f9:ef > (V) 00:50:f2:cd:ce:04,

ethertype IPv4 (0x0800),length (VI) 62: (II)192.168.2.38. (III)445 >

(I)192.168.2.37. (III) 4467: (IV)S 2672924111:2672924111(0) (IV) ack

1052151846 win 64240 <mss 1460,nop,nop,sackOK>
```

Network captures prove useful for a lot of network-level analysis. However, as soon as applications need to be analyzed, the packet captures do not provide the necessary application logic to reproduce application behavior. In those cases, other sources are considered.

1.4 Information Visualization Process

Visualization of data is not always a straightforward process. It is important that the problem or objective is very clear to start with. After the problem has been clearly identified, various decisions must be made. Color assignment, for example, is one choice to make, and the choice of the right graph for the data and problem at hand is another.

To increase the chances of generating a useful graph that addresses the objective and helps solve the problem, An information visualization process is introduced inspired by the InfoVis [14] developed by Yuri Engelhardt and Juan C. Dursteler. Figure-1.3 illustrates the six steps of the information visualization process. Starting from the left, the problem needs to be identified first [1].

Based on the problem statement, specific data sources are needed to answer the questions posed in the problem statement. After the data has been identified, various processing steps are applied to eventually arrive at a graphical representation. This process is the same for any type of data analysis and is not specific to a security use-case. The process is security specific in the decisions that are made along the process.

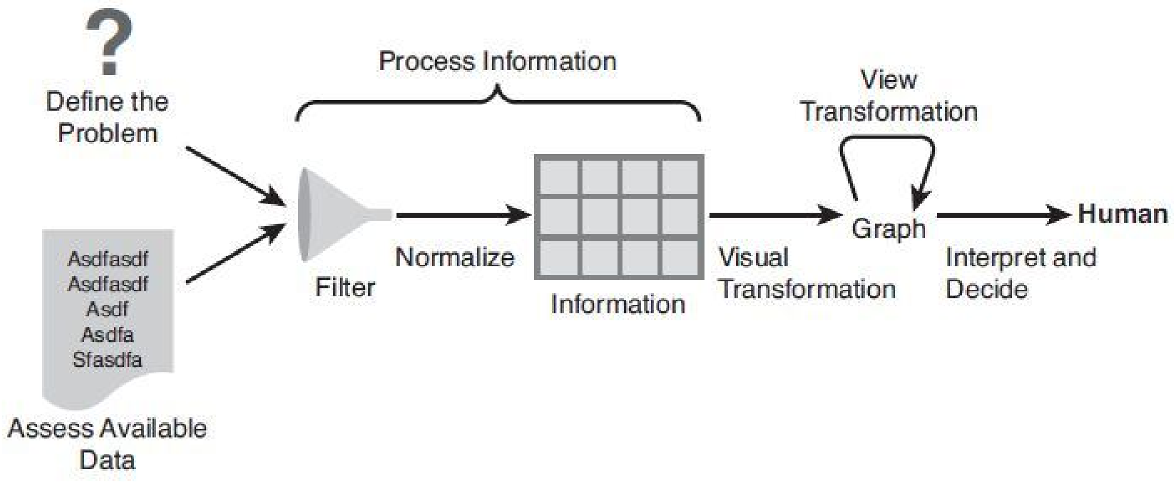


Figure 1.3 : Information visualization process [1]

Step 1: Define the Problem

Visualization should never be data driven .Visualization should be use-case driven . It should be known what is that need to understand, communicate ,or explore? What is expected to see? This is why the very first step in the information visualization process is about defining the problem.

Step 2: Assess Available Data

The next logical step after defining the problem is to find out what type of data that need to potentially answer the questions posed. What data is available? What logs files could help answer the problem(s) stated in Step 1. Is there additional data that is needed apart from log files? For example, is there a mapping from IP addresses to geographic locations?

Step 3: Process the Information

At this point, it is known what the problem is that is to solve and have identified and collected all the data necessary to do so. However, the data is not in the right format to go ahead and use visualization tools to generate graphical representations .There is need to process the data first and bring it into the right format. In other words, transforming the data into information .Transforming the data into information adds meta data to the fields that explains the meaning or semantics of the data. This transformation process is also called parsing.

The basic idea of a parser is that it takes a raw log file and extracts the individual fields from it.

Step 4: Visual Transformation

The output from the previous steps are parsed log entries that contain all the information that need for visualization. This includes things such as asset role mappings, geographic locations, or DNS hostnames. The parsed output should be available in a comma-separated format, a CSV file. CSV makes it simple to further process and work with the data. The next step in the process is mapping the data into some visual structure that produces a graphic representation. The transformations that make this possible are called visual transformations. For example, a three-column input file can be transformed into a 3D scatter plot, using each of the input data fields as one of the graph dimensions. The same input data could also be used to produce a 2D scatter plot with the third input field representing size or color of the data points.

Step 5: View Transformation

In Step 4, it produced a first visual representation of data. The tool used to generate the graph can modify choices of color, shape, size, graph type, and so on. Even if this is not the case, it can still restart the process and iteratively use the knowledge that is gained. Sometimes it is sufficient to zoom in on a specific area and highlight that data. At other times, need to go back to problem and start over.

Even if one gone back and regenerate graphs, the first graph provides some initial insight into the log file. It provides context and helps determine which log entries or graph components should be filtered out to highlight and identify the really interesting parts of the log file. This is generally an iterative process of generating multiple graphs with varying filters. However, one should be cautious when filtering nodes or log entries so that it don’t inadvertently remove critical data.

Step 6: Interpret and Decide

Step 5 potentially resulted in multiple graphs to focus the graph on the relevant data. The final graph is the one that, can now be used to satisfy initial objectives. This might seem to be the easiest step in the entire process, to read the final graph. However, it often turns out to be not so trivial.

A method that often proves useful when analyzing and interpreting graphs is the comparison against other graphs. These graphs can either be ones that have been generated over the same type of data but at a previous point in time or it can be a reference graph for that type of visualization. If it is known that a graph should always display a certain pattern and it suddenly does not show up anymore, it has just found an anomaly [1].

Chapter 2

Literature Review

Nowadays, many organizations heavily rely on computer networks which then become critical parts of their infrastructures. At the same time, the number of malicious attacks against such networks is exploding . In order to guarantee the security and the stability of the networks they manage, system administrators must monitor them efficiently. That monitoring task can be done displaying selected information about the network traffic in a textual form and reading this output directly. Unfortunately, the amount of data captured on the network they need to analyze to perform that task tends to become quite huge and that method is no longer really feasible. Various tools exist to help system administrators dealing with this issue. They are mainly based on automated systems (learning techniques, signature databases or statistical analysis) [15].

Recently, visualization techniques also started to be used for the monitoring task. The idea is to take advantage of human visual processing and pattern recognition that are rather unexploited resources in traditional tools. Most of these tools are built around 2D visualization zones. Some try to exploit 3D as it permits to create more complex representations, however they generally face clarity and usability issues [16].

There are number of tools in the visualization area that have applied on the network data visualization. Commonly, network security data monitoring is the part that most of the visualization applications have focused on more compared with others. Information on malicious attacks that have been triggered on an abnormal detection device will be presented to the network administrator [17].

There are some other areas that visualization tools have focused on such as network intrusion detection. Before discussing these existing tools it is required to know about data inputs or data files. Different visualization tools use different types of data files.

2.1 Data Input Formats

The visualization tools cannot directly read data from security devices. They will not understand, for example, a firewall log. The most common and important file formats that needed to know are comma separated value (CSV), the TM3 format, GraphViz's DOT format, Graph Markup Language (GML) and SNMP data.

2.1.1 CSV

A comma separated values, CSV, file stores fields or data dimensions separated by a comma. The following shows an example:

```
192.168.0.5,192.168.0.18,23

10.2.3.1,10.0.0.2,2912

192.168.1.3,192.168.34.78,53
```

There are three columns or data dimensions. The first column represents the source address, the second column the destination address, and the last column the destination port. It cannot derive this from the file alone. In some cases, the tools expect a header line that gives a name to each of the columns. Other tools don't respect such a line and assume that the first line is also part of the data[18].

Suppose, if there is need to represent a value in a column that contains a comma itself for example, consider the string 80,443,993.,wanting to keep this string as a single value and not break it up into individual columns. In that case, there is need to quote the string, which will look like this: "80,443,993". Most programs can understand this format.

Some tools support a TSV, a tab separated values format. This is nothing other than a CSV file where, instead of commas, tabulators are used.

2.1.2 TM3

The TM3 file format is a bit more complex, but also more expressive, than the very common CSV files. The main benefit of TM3 files is that a data type is assigned to each of the data columns. This is done by utilizing header lines. The first two lines in the file are dedicated headers. The first line defines the name of the column, which is purely cosmetic, and the second line defines the data type for the column. Generally, the accepted values are STRING, INTEGER, FLOAT, and DATE. Tabulators separate columns. Here is a simple example:

```
Name Age DOB

STRING INTEGER DATE

Nitin Verma 21 10/09/1986
```

Quotes can be used to surround fields, but are optional. The data types have the advantage that the tool reading the file knows what type of data to expect. This has an influence on sorting, operations that can be applied to the fields, and the way the field is displayed [1].

2.1.3 DOT

The DOT attributed graph language is what its name suggests: a simple language that defines a link graph. Each node and each edge has an entry in this file that describes exactly how the nodes and edges have to be represented. Attributes such as color, size, shape, and so on can be specified. In addition, global options can be specified. These are applied to the entire graph. Things such as pagination, size of the drawing, and so forth are specified as global options. Depending on what layout algorithm used in GraphViz[19], the parameters that can be used for nodes and edges, as well as the global graph options, are slightly different. The main ones stay the same, but options specific to the layout algorithms vary. This property of DOT files breaks the paradigm of separating data from presentation. DOT files are not pure data input files, unlike GML files. The following is a simple DOT file that illustrates the structure and expressiveness of the file format:

```
digraph structs {

graph [label="AfterGlow 1.5.8", fontsize=8];

node [shape=ellipse, style=filled, fontsize=10, fillcolor=green];

edge [len=1.6];

"nit" -> "Busy";

"ver" -> "Encryption";

"nit" [fillcolor=white, label="Nitin", shape=box];

"ver" [fillcolor=blue, label="Verma", shape=box];

}
```

This example first defines the global properties for generating the graph. The graph keyword indicates global settings for the graph, the label in this case. These properties are followed by default values for node and edge properties. If a node or edge definition is not specifying anything different, these are the settings that are used. The example then defines the edges in the graph. There are only two edges. Following the edge definitions, special properties are defined for two of the nodes. The color, label, and shape for these two nodes are not the default ones, but the ones specified. Figure 2.1 shows how GraphViz renders the example DOT file [20].

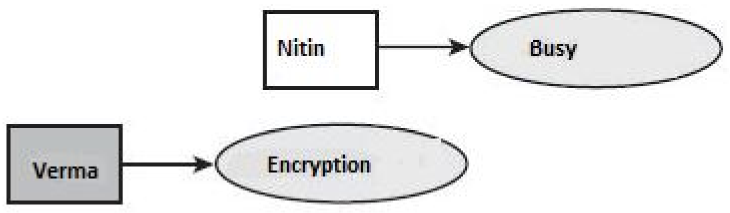


Figure 2.1 : Example of a DOT file shown by GraphViz

2.1.4 GML

One way of representing a link graph is by using a DOT file. A number of visualization tools do not support DOT files but are using a very similar format, the Graph Modeling Language (GML). This format is similar to a DOT file. It consists of a hierarchical key value list. The following is a simple example graph [1]:

```
graph [

comment "This is a sample graph"

directed 1

IsPlanar 1

node [

id 1

label "Node 1"

]

node [

id 2

label "Node 2"

]

edge [

source 1

target 2

label "Edge from node 1 to node 2"

]
```

A graph has some global parameters. Those are followed by a definition of all the nodes. The nodes are required to contain an id. Following the node definitions, all the edges are defined by using the source and target keys. The IDs for source and target are the IDs used in the nodes. Additional properties can be added to nodes and edges in key-value form. It is up to the visualization tools to interpret the additional keys [21].

2.2 SNMP

Simple Network Management protocol (SNMP) is one of the most commonly used technologies when it comes to network visualization. SNMP is based on the manager/agent model consisting of an SNMP manager, an SNMP agent, a database of management information, managed SNMP devices and the network protocol. The SNMP manager provides the interface between the human network manager and the management system. The SNMP agent provides the interface between the manager and the physical device(s) being managed [22].

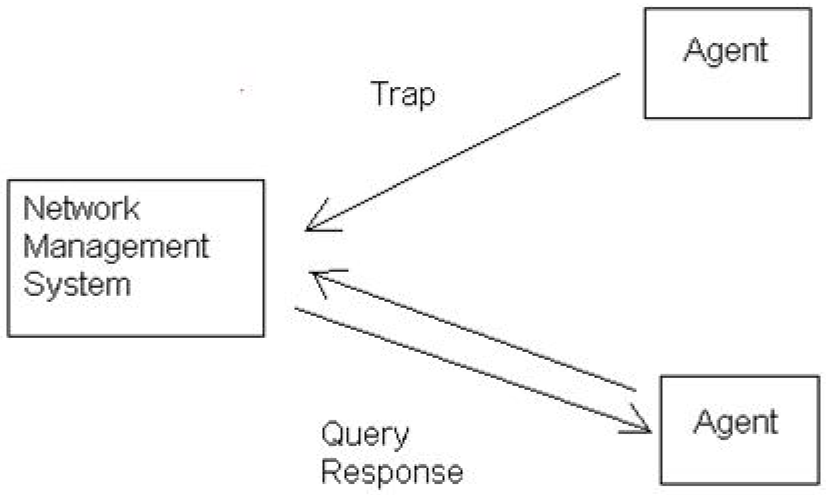


Figure 2.2 : Network Management using SNMP

The SNMP manager and agent use an SNMP Management Information Base (MIB) and a relatively small set of commands to exchange information. The SNMP MIB is organized in a tree structure with individual variables, such as point status or description, being represented as leaves on the branches.

A MIB file is just ASCII text, so it can be viewed in any word processor or text editor, such as Microsoft Notepad. Some manufacturers provide precompiled MIBs in binary format, but those aren't readable. The raw ASCII version of the MIB file is required.

The MIB is written in ASN.1 notation. (The initials stand for Abstract Syntax Notation 1.) ASN.1 is a standard notation maintained by the ISO (International Organization for Standardization) and used in everything from the World Wide Web to aviation control systems.

Each SNMP element manages specific objects with each object having specific characteristics. Each object characteristic has a unique object identifier (OID) consisting of numbers separated by decimal points (i.e., 1.3.6.1.4.1.2682.1). These object identifiers naturally form a tree . The MIB associates each OID with a readable label and various other parameters related to the object. The MIB then serves as a data dictionary or code book that is used to assemble and interpret SNMP messages [23].

When an SNMP manager wants to know the value of an object / characteristic, such as the state of an alarm point, the system name, or the element uptime, it will assemble a GET packet that includes the OID for each object / characteristic of interest. The element receives the request and looks up each OID in its code book (MIB). If the OID is found (the object is managed by the element), a response packet is assembled and sent with the current value of the object / characteristic included. If the OID is not found, a special error response is sent that identifies the unmanaged object.

When an element sends a TRAP packet, it can include OID and value information (bindings) to clarify the event. Remote client sends a comprehensive set of bindings with each TRAP to maintain traditional telemetry event visibility. Well-designed SNMP managers can use the bindings to correlate and manage the events. SNMP managers will also generally display the readable labels to facilitate user understanding and decision-making.

2.3 Available Visualization Tools

Many tools are available in visualization area. These tools have their strengths, weaknesses, and generic capabilities. All the tools to be covered here are tools that take actual data and not coordinates to create the graphs. There are other tools in that have to feed coordinates to generate a graph. There are two basic types of graphing and visualization tools: ones that generate static data graphs and others are stand-alone applications. Static data generation tools are ones that can be scripted and use on the command line to generate image files, such as GIF or SVG. They do not provide a viewer, nor do they provide any interactivity to manipulate or navigate the generated graph. This is in contrast to stand-alone applications, which have a user interface that offers varying degrees of capabilities to interact with and manipulate the graphical output.

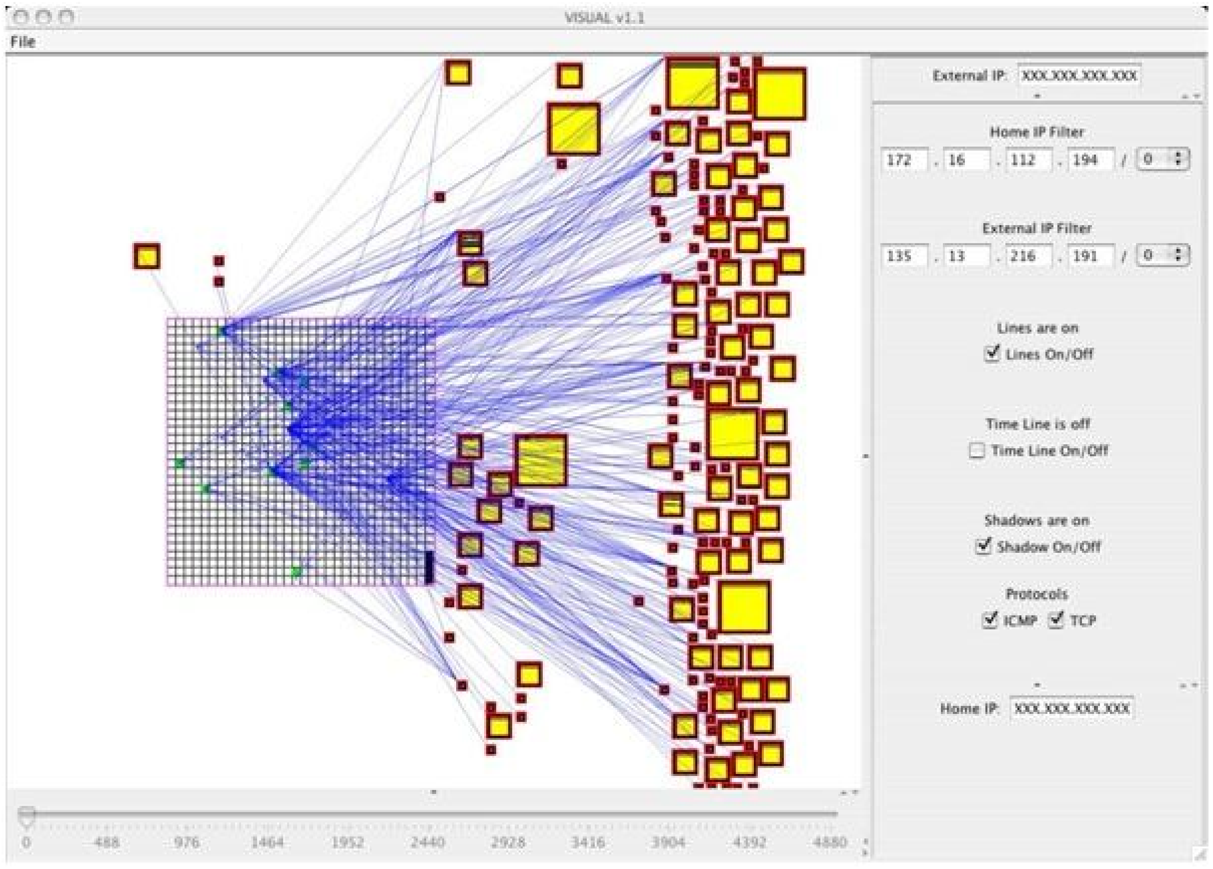
2.3.1 Stand-Alone Applications

Stand-alone visualization applications do not just visualize a certain dataset. They also display the output and let the user interact with it. Interaction is crucial for data exploration. The applications presented here offer different levels of interaction. Some offer only the most basic operations, such as zooming. Others offer full interactivity, with brushing, filtering, and even graph editing.

2.3.1.1 Visual Information Security Utility (VISUAL)

In order to rapidly aware with the security conditions of their network, Visual Information Security Utility for Administration Live (VISUAL) is a network security visualization tool that allows network administrators to examine the communication networks between internal and external hosts [24]. VISUAL applied the concept of dividing network space into a local and network address space and a remote network address space. In order to produce its data visualizations, data will be taken from the log files of Tcpdump or Wireshark. It provides a computer security visualization that gives a rapid perception and temporal visualization of traffic .

The advantage of VISUAL is to provide a quick overview of the current and recent communication patterns among the monitored network. Administrators can specify their network and remote IP by using home and remote IP filter. Based on the information provided by IP filter, administrators can identify any single external hosts that are connected with the number of internal hosts from a grid, which may be relevant to be used in their network [24].



External IP: XXX.XXX.XXX.XXX

Home IP Filter
172 . 16 . 112 . 194 / 0

External IP Filter
135 . 13 . 216 . 191 / 0

Lines are on
✓ Lines On/Off

Time Line is off
☐ Time Line On/Off

Shadows are on
✓ Shadow On/Off

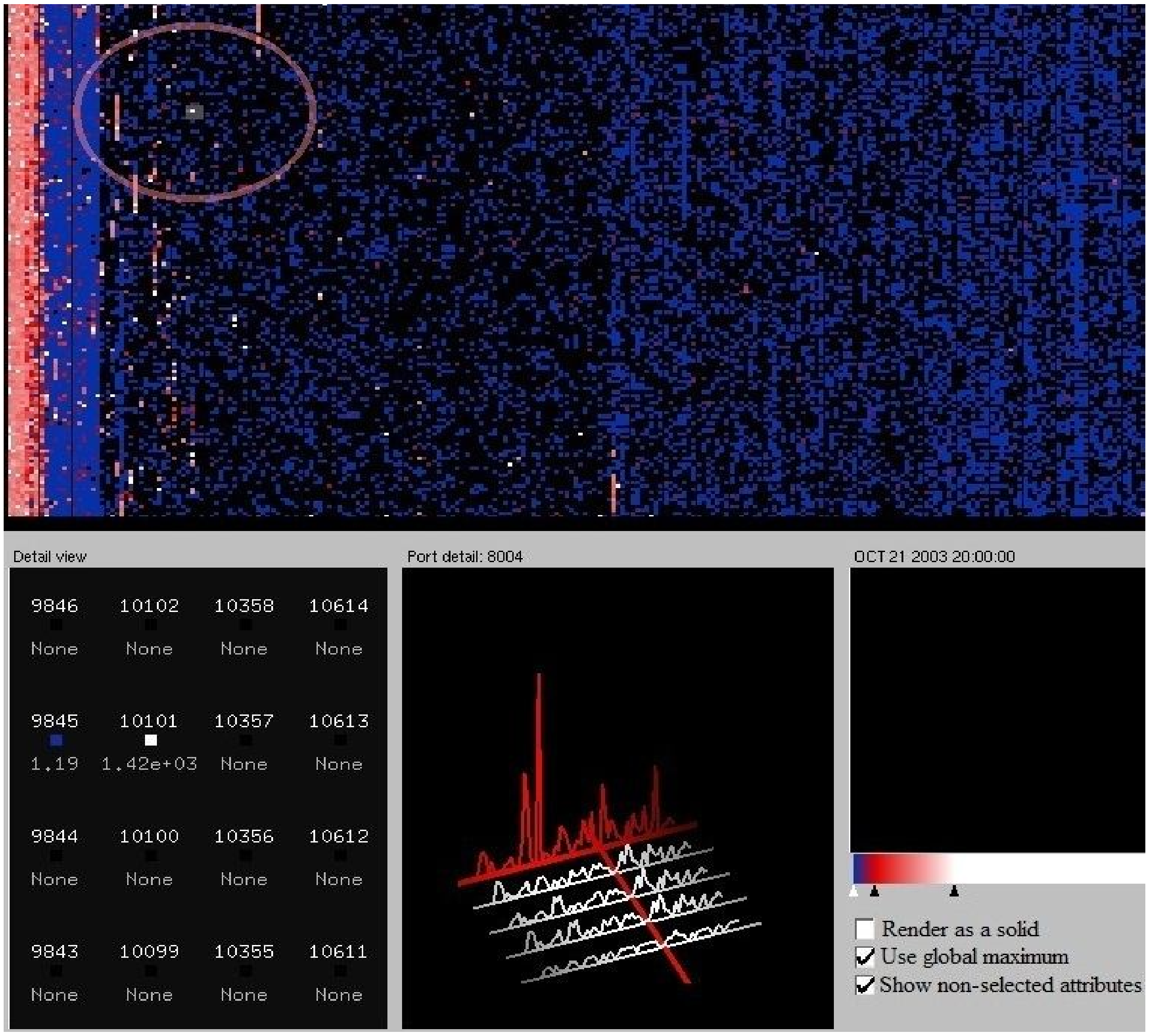
Protocols
✓ ICMP ✓ TCP

Home IP: XXX.XXX.XXX.XXX

Figure 2.3 :VISUAL allowing communication among hosts [24]

2.3.1.2 PortVis

Another network security visualization tool is PortVis [25]. It focuses on a single host at a time and doing the analyzing on it. The main purpose of this tool is to present outside data entities to outside security specialists. Information such as each TCP port during a period of one hour is being visualised and large scale of security occurrence will be detected by PortVis. PortVis also allow for small scale security occurrence detection, which allowed for further investigation.



Detail view

9846	10102	10358	10614
None	None	None	None
9845	10101	10357	10613
1.19	1.42e+03	None	None
9844	10100	10356	10612
None	None	None	None
9843	10099	10355	10611
None	None	None	None

Port detail: 8004

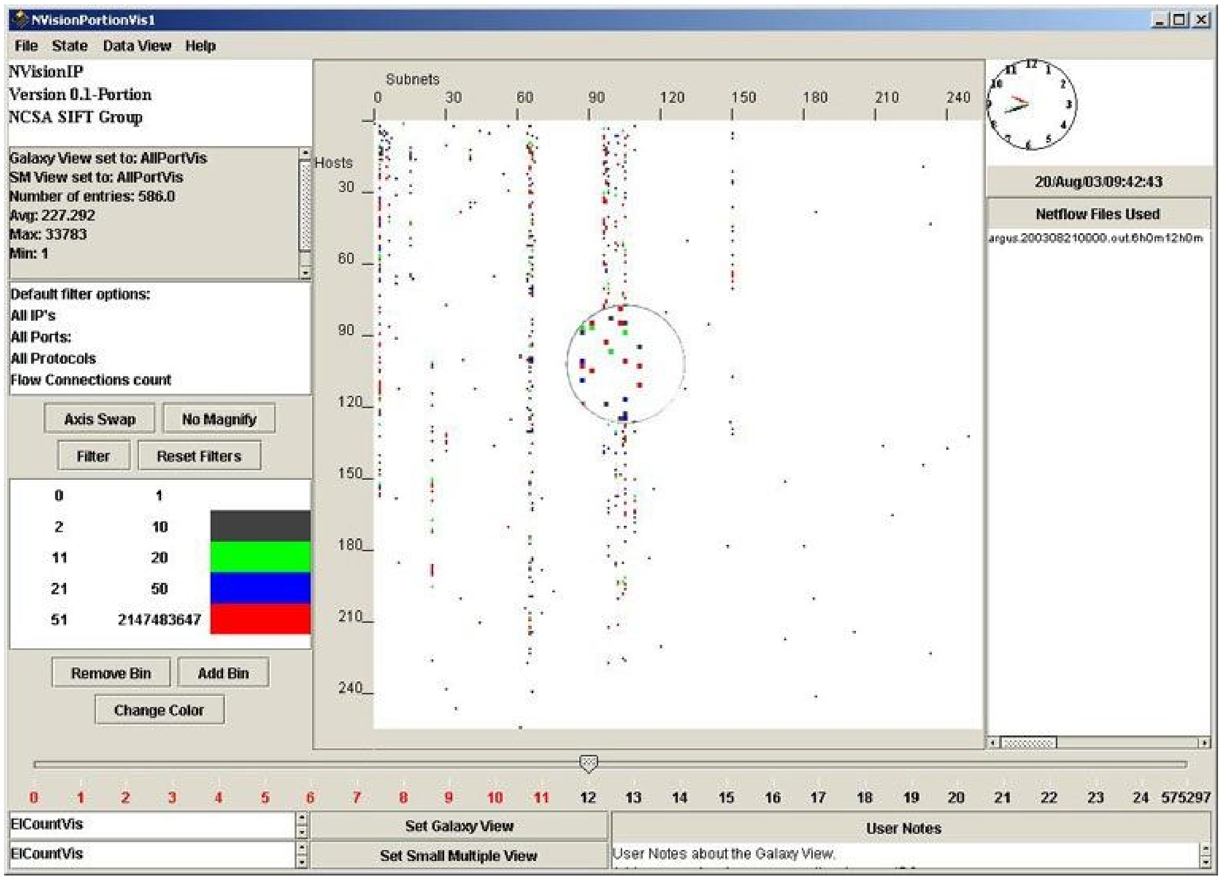
OCT 21 2003 20:00:00

☐ Render as a solid
✓ Use global maximum
✓ Show non-selected attributes

Figure 2.4 :PortVis- Basic summary of information from each port [25]

2.3.1.3 NVisionIP

Besides that, Figure-2.5 shows the NVisionIP [26] is also a visualization tool that targeted to provide and improve the overall situational awareness of the network among network security administrators. A graphical representation of a class-B network and numbers of different views of the data will be presented to network security administrators. There are three main visualization views in a single application of NVisionIP, namely Galaxy, Small Multiple and Machine visualization views. NVisionIP targeted to improve the interactivity among this visualization views by allowing them to transferring data from one visualization views to other visualization views.



NVisionIP
Version 0.1-Portion
NCSA SIFT Group

Galaxy View set to: AllPortVis
SM View set to: AllPortVis
Number of entries: 586.0
Avg: 227.292
Max: 33783
Min: 1

Default filter options:
All IP's
All Ports:
All Protocols
Flow Connections count

0	1	
2	10	⬛
11	20	🟩
21	50	🟦
51	2147483647	🟥

20/Aug/03/09:42:43

argus.200308210000.out.6h0m12h0m

ElCountVis

ElCountVis

User Notes about the Galaxy View.

Figure 2.5 : NVisionIP- improving situational awareness [26]

2.3.1.4 Multi Router Traffic Grapher

The Multi Router Traffic Grapher (MRTG) [27] is a tool to monitor the traffic load on network-links. MRTG generates HTML pages containing PNG images which provide an almost live visual representation of this traffic. It was originally developed by Tobias Oetiker and Dave Rand to monitor router traffic, but has developed into a tool that can create graphs and statistics for almost anything. MRTG is written in Perl and comes with full source. It can run on Windows, Linux, Unix, Mac OS and NetWare. MRTG is free software licensed under the terms of the GNU General Public License.

MRTG uses a highly portable SNMP implementation written entirely in Perl . There is no need to install any external SNMP package. SNMP queries are used on a regular interval to generate graphs. External readers for MRTG graphs can create other interpretation of data [27]. MRTG can be used not only to measure network traffic on interfaces, but also build graphs of anything that has an equivalent SNMP MIB - like CPU load, disk availability, temperature, etc.

Issues with MRTG

- MRTG generates each graph in every 5 minutes, if there are hundreds of graphs it causes a lot of overhead.
- It also has very few customizable graphing options.
- On the graphing front, MRTG graphs always use a Y axis starting at 0, if only want to see the relevant values in a range (for temps it might only want to graph from 50F to 90F) it may have problem
- Disk space is always an issue.
- MRTG management itself can be tedious work.

2.3.1.5 Round Robin Database Tool

RRD tool [28] fills in the gaps that MRTG leaves wanting, and provides for open customization that was difficult if not impossible before. RRD tool is the open source industry standard, high performance data logging and graphing system for time series data. Tobi Oetiker wrote RRD tool as a replacement for MRTG and licenses it as free software under the terms of the GNU General Public License (GPL).It can write custom monitoring shell scripts or create whole applications using its Perl, Python or PHP bindings.

RRD tool assumes time-variable data in intervals of a certain length .The main goal of RRD tool is the creation of time graphs that display the trend over time of one or more values. A time graph displays on the x axis the time and on the y axis the values of the variables. The variable values are collected by SNMP objects or round trip time values results of ping request etc.

The specific of RRD tool is that it's meant for data that changes all the time, continuously. There's a lot of data in a computer system that can be collected in this manner. Processor temperature, how much data to transfer via network connections, how much RAM or other resources consume at any given time, disk space and activity, ping response times and so on. If it can get Figured , it can be plotted.

The number of records in a ".rrd" file never increases, meaning that old records are frequently removed so no overhead of huge amount of data.

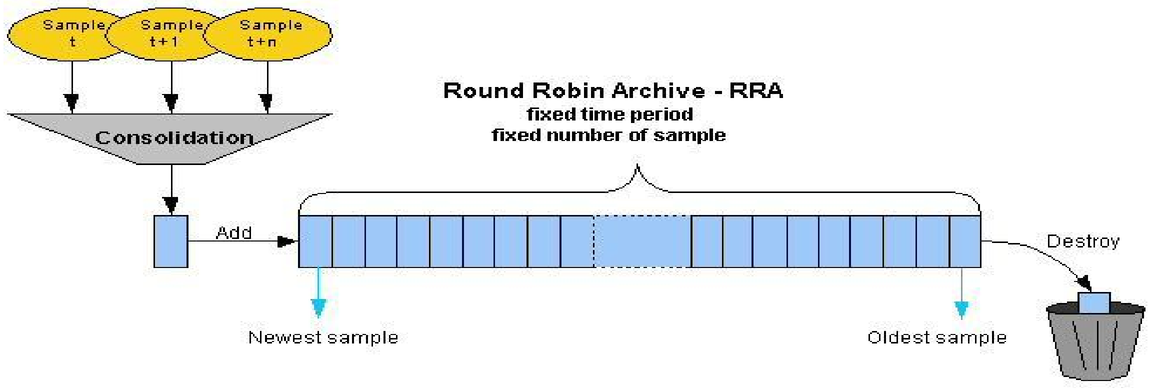


Figure 2.6 : Round Robin Archive

2.3.2 Static data generation tools

This section introduces some visualization tools that generate static graphs as output. In contrast to interactive, stand-alone applications, these tools mostly come in the form of scripts or tools launched on the command line. The output is generally written to an image file. The disadvantage of these tools is that it cannot interact with the visual output to interactively explore the content. On the other hand, these tools can be scripted and be used in processes to automatically generate graphical output.

Often, the data that is needed to visualize is available in CSV format or may be stored in a database. However, some of the tools require more complicated data input formats (for example, a DOT file). To bridge the gap between a simple CSV file and a DOT file, AfterGlow can be used. It is not a classical visualization tool, but it helps translate a CSV file into a DOT file, based on a set of configuration parameters.

2.3.2.1 AfterGlow

AfterGlow is a collection of scripts that facilitate the process of generating link graphs. The tool is written in Perl and needs to be invoked via the command line. As input, AfterGlow expects two- or three-column CSV files, and it outputs a DOT file, or it can generate input for the large graphing library . Among visualization tools, a number of them use DOT files as input [29].

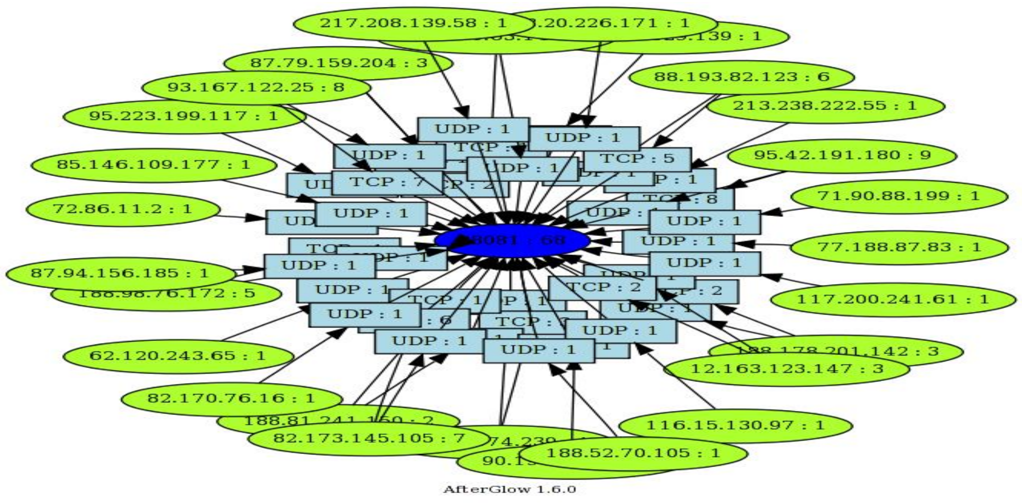


Figure 2.7 : AfterGlow Visualization

Generating those files manually is not easy. AfterGlow bridges this gap and enables users to use simple CSV files to generate their DOT graph descriptions.

To define all of these properties, AfterGlow uses a property file that is passed as an argument on the command line. The property file consists of a set of assignments.

2.3.1.2 GraphViz

When it comes to generating link graphs, this is one of the better tools available. The GraphViz package consists of tools that translate a description of a graph into an image. The output format can be anything from GIF to SVG. However, GraphViz is a scripted tool, and the output is static. GraphViz ships with two tools that can be used as stand-alone applications [19].

GraphViz package implements a variety of layout algorithms for link graphs. Each algorithm is represented by a different tool that can generate the graph output:

- **dot** generates hierarchical layouts.
- **neato** is used to draw graphs by using a spring model for computing the layout.
- **twop**i draws graphs using a radial layout.
- **circo** draws graphs using a circular layout. It generally generates very big graphs that are not very useful.
- **fdp** uses a different spring model to layout the graphs.

All the tools read DOT attributed graph language files. The format is fairly simple and is documented. Each tool has a variety of parameters it understands. Here is a sample way of rendering a graph:

```
cat input.dot | neato -Tgif -o output.gif
```

This command instructs neato to render the input from the file *input.dot* and create a GIF image.

2.4 Comparison of various Visualization tools

A brief comparison of all the tools discussed so far, is shown in table-2.1.This comparison tries to differentiae among various important aspects and properties of these tools.

Name	***Open Source ?***	***Written in Language***	***Platform needed***	***Input data format***	***Output data***
VISUAL	No	N/A	Windows, Unix	PCAP	Visualization among hosts
NVisionIP	Yes	Java	Unix , Mac OS X	Textual Argus Data	Customizable packet visualization
MRTG and RRDtool	Yes	Perl	Unix, Windows, Mac OS X	SNMP Queries	Customizable histograms
Afterglow	Yes	Perl	Perl	CSV	DOT
GraphViz	Yes	Perl, Python etc.	Linux, Windows, Mac OS X	DOT	Link graph Layouts

Table 2.1 : Comparison of visualization tools

The most important properties are shown in the columns . This comparison summarizes all the tools at a glance and shows the input and output behavior of these different tools. This table can also help the user to take decision while choosing the appropriate tool according to requirements.

2.5 Visualization Methodologies

E. Wes Bethel et al. [4] presented an interdisciplinary combination of scientific data management and visualization technologies targeted at reducing the time required for data filtering, querying, hypothesis testing and knowledge discovery in the domain of network connection data analysis .They contributed objective measurement of how using state-of-the-art index/query technology can reduce the filtering or data mining portion of the analytics duty cycle and an approach to knowledge discovery that relies on multi resolution queries and statistics from user-defined multidimensional range queries presented as histograms to perform interactive analysis .

Doris Wong Hooi Ten et al. presented a visualization methodology architecture [30] for network monitoring , which focuses on its second module in the architecture namely, analysis and intelligence module. Generally, visualization tends to be an iteration process. There are two agents in its first data procurement module to solve the system overhead problems in data searching and data capturing process. Data mining technique such as association rules (defined policy) is applied in the analysis and intelligence module to verify the usefulness of the tool [31] .

It is very important task to scan the whole network for discovering malicious activities. Once the whole network is scanned ,then we can find which systems are becoming threat to the whole network. On the basis of this concept Zhang Jiawan et al. [33] proposed a novel approach for efficient network scans detection ,where a visual interactive network scans detection system called ScanViewer is designed to represent traffic activities that reside in network flows and their patterns. The ScanViewer combines the characteristics of network scan with novel visual structures, and utilizes a set of different visual concepts to map the collected datagram to the graphs that emphasize their patterns.

An *expert-aware approach* proposed by Wong et al. [32] used to design a system which formulated with a large amount of high dimensional network data and adaptive for different types of users. In the preliminary phase .A survey was conducted among different types of computer users and collected data from them. Computer users provided their requirement of network data [32]. The system will learn from continual user feedbacks in order to statistically satisfy the needs of majority tool users. The expert-aware approach looks at the users’ expertise level in network security and adapts the most comprehensive visualization views that are best suitable for the user.

The architecture is mostly based on the node concept. A node is an entity containing several elements such as, an icon (type depends on the programming language used), an x coordinates and a y coordinates as an Integer type (to localize the icon in the scene), some Strings containing the different IP addresses, a date type and also a list of nodes.

In this whole chapter various application, tools and methodologies in perspective of network data visualization are discussed. Standalone applications like MRTG [27] and RRD tool [28] are used to create any type of graphs on the basis of SNMP query data. VISUAL [24] tool is used to show communication among internal and external hosts on a particular IP range. Moreover , PortVis [25] is used to analyze single host activities at a time , activities on different ports of the host are being visualized. On the other hand NVisionIP [26] attempts to improve situational awareness of the user by showing different views according to need.

Static data generation tools like AfterGlow [29] and GraphViz [19] come in the form of scripts and can be executed using command prompt. AfterGlow takes CSV file as a input and produces a graph language file or DOT file. On the other hand GraphViz needs DOT as a input and through using various layouts results the output graph. The brief comparison among all these tools is also presented which tries to differentiates them on the basis of functionality ,platform ,output etc.

In the next chapter the problem and the concise objectives of the thesis are listed.

Chapter 3

Problem Statement

Network Traffic logs can be voluminous, a capable network with 1000 nodes usually produces Gigabytes of log file data and to find information like which IP address is sending malicious data is a huge task.

Network Visualization and Analysis helps administrators to take reliable and faster decisions (eg. Blocking a particular IP or Port).

Objectives :

Followings are the objectives that are aimed to be achieved during entire thesis-

I. To study and explore various network visualization tools with the help of local area network .

II. To Design and develop network visualization framework for Thapar University.

III. To demonstrate the use of developed framework by creating visualization data of Cyberoam captured log files.

Chapter 4

Implementation details and Results

4.1 Implementation Setup to understand visualization

It is very essential to understand how actually network traffic visualization is performed. Therefore, a real time experimental setup is established to make graphs of traffic data. The actual setup is shown in Figure -4.1.

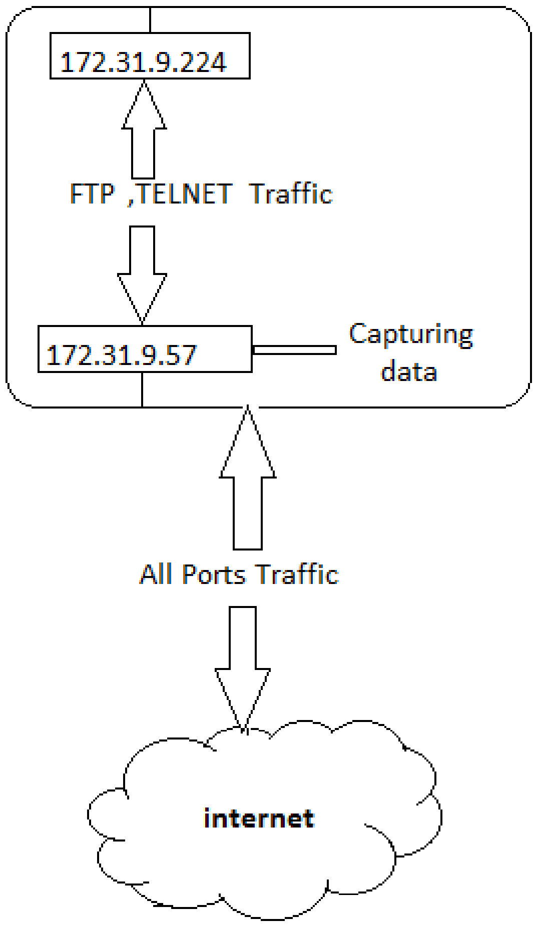


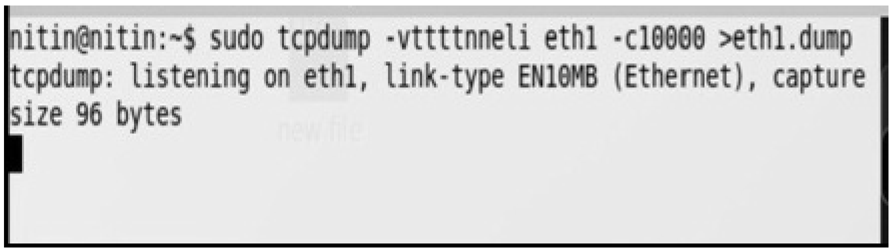
Figure 4.1 : Experimental Setup

As , it can be seen in diagram that both the PCs with IP 172.31.9.57 and 172.31.9.224 are connected to the internet. Network traffic data is captured from the interface of PC of IP 172.31.9.57 .All the traffic coming from and going to internet cloud is captured .Meanwhile, another PC is busy with performing Telnet and FTP sessions , so that other data together with HTTP data is also captured .Thus all traffic is passing from the interface of 172.31.9.57 is captured for visualization purpose.

4.1.1 Capturing network traffic using tcpdump

A variety of tools provide facility for capturing the packets like wireshark , tcpdump , tshrak etc. Among these packet capture tools , tcpdump is chosen because it is an extremely useful network packet tracing system. While not as feature-rich as programs such as Wireshark, its packet “dump” output can be used as input by other programs to analyze the dump. In a pinch, and for network debugging, tcpdump works good.

Therefore, tcpdump is used to capture the packets for visualization of the traffic. Required fields of the tcpdump can be collected through passing the various parameter and options available in it. For visualization purpose we passed the following parameters in tcpdump , as shown in Snapshot-4.1 .



```
nitin@nitin:~$ sudo tcpdump -vttttnneli eth1 -c10000 >eth1.dump
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture
size 96 bytes
```

Snapshot-4.1 : Packet capture using tcpdump

The significance of each option passed is described as below -

-v Displays verbose output. For example the time to live and type of service information in a packet displayed.

-ttt Print a timestamp in default format proceeded by date on each dump line .

-nn Does not convert addresses to names and does not display domain name qualification of host names.

-e Print the link-level header on each dump line.

-l Buffers the stdout line. This is useful if we want to see the data while capturing it.

-i Listen on interface. If unspecified, tcpdump searches the system interface list for the lowest numbered, configured up interface (excluding loopback).

-c Exit after receiving *count* packets.

In this case **`eth1`** interface of PC with IP 172.31.9.57 has been chosen for packet capturing. All the traffic passing from this interface will be captured. Moreover , tcpdump will exit after receiving 10000 packets.

4.1.2 Selecting approach for traffic visualization

Till now all the traffic that is incoming and outgoing from the interface `(eth1)` is captured .Now there is need to visualize and analyze it .Earlier , in chapter -2 many visualization methodologies and approaches were discussed . Among all those there is need to choose the most appropriate approach for creating graphs. The basic approach of creating graphs is by converting tcpdump file into CSV format . The whole approach can be represented pictorially .

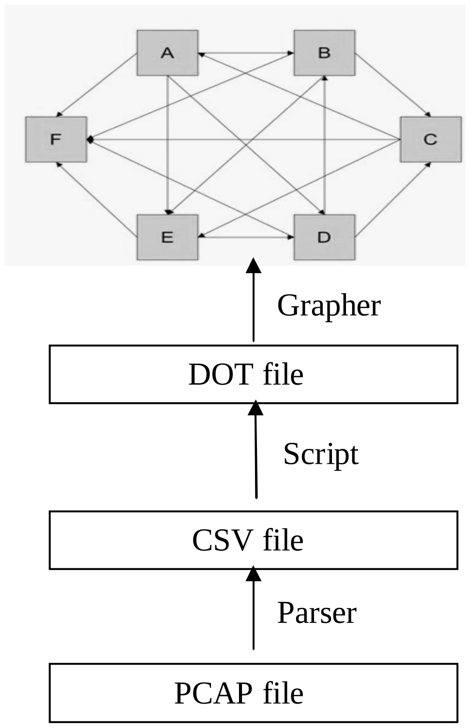
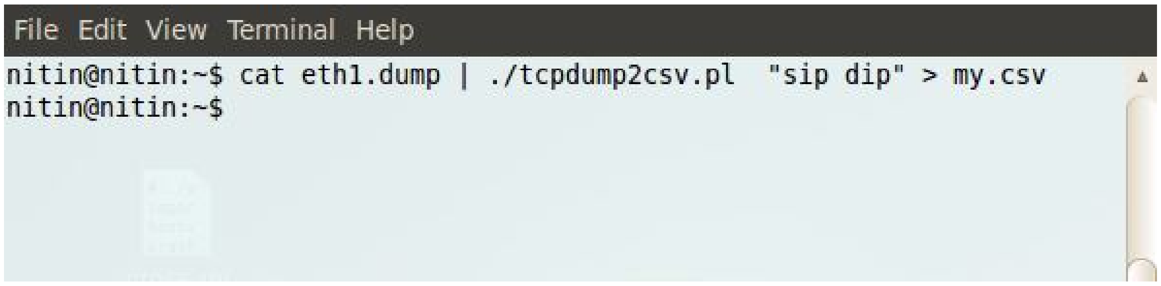


Figure – 4.2 : Graph creating process

As PCAP file (tcpdump file) is already captured. Now it is required to convert it into CSV (Comma Separated Values) file .For converting into CSV file , any parser can be used. As shown in the Snapshot-4.2 a parser `tcpdump2csv.pl` is used. This parser can take tcpdump file as input and parse this dump file. After parsing it creates CSV file as output.



```
nitin@nitin:~$ cat eth1.dump | ./tcpdump2csv.pl  "sip dip" > my.csv
nitin@nitin:~$
```

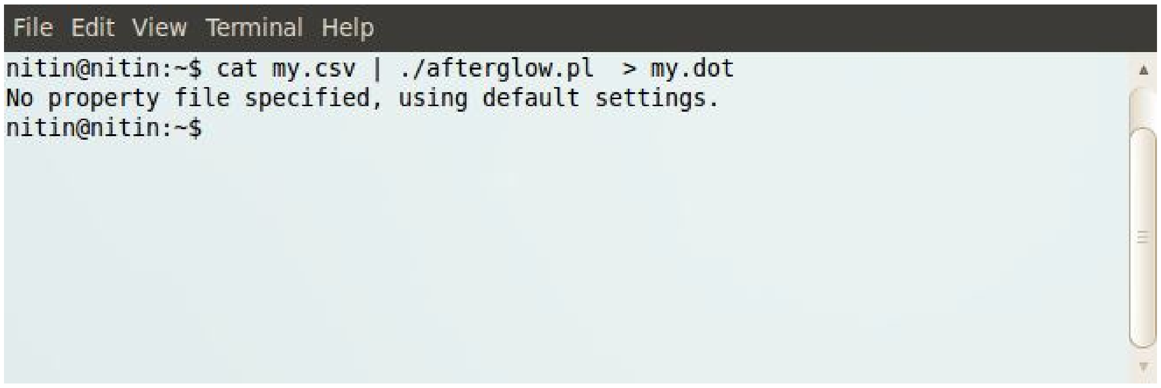
Snapshot – 4.2 : parsing tcpdump file into CSV

As shown in the Snapshot for parsing the tcpdump file, two fields are passed `sip` and `dip` that mean Source IP and Destination IP will be used for further visualization. Comma Separated Values are created according to need of visualization. This way the visualization of the connection among all the Source IP and Destination IP is possible. Moreover, other different fields can also be passed ,if there is need to visualize them. Other important possible fields can be-

sip	Source IP
dip	Destination IP
ttl	Time To Live
sourcemac	Source Mac address
destmac	Destination Mac address
sport	Source Port
dport	Destination Port

Any fields that are needed to visualize are subjected to be in Comma Separated Values. Further they can be passed to any other script for visualization purpose. Here the source IPs and destination IPs are stored in `my.csv` which are separated by commas.

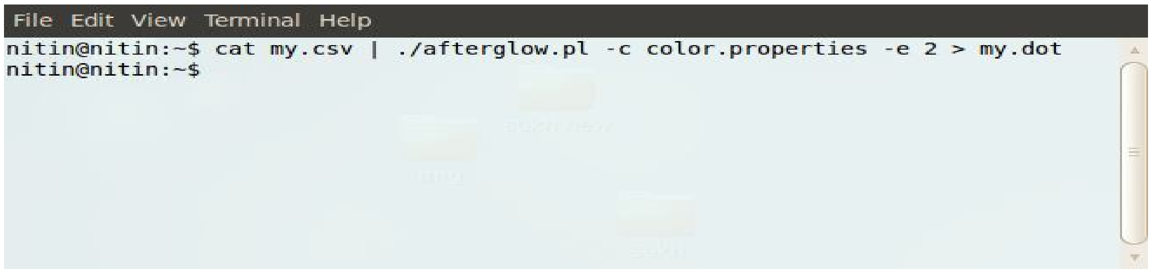
After getting CSV file, there is need to commute it into graph language file or DOT file. Afterglow.pl [29] can be used to convert into graph language file .In the following Snapshot `my.csv` is converted into `my.dot.`



```
nitin@nitin:~$ cat my.csv | ./afterglow.pl  > my.dot
No property file specified, using default settings.
nitin@nitin:~$
```

Snapshot – 4.3 :Converting CSV file into DOT file

As shown by above command , only default settings are used for creating DOT file. It means that properties of afterglow script are not used. These properties are used to configure color output or color coding for different need of visualization ,like for particular IP range, particular port range or specific ports only. `color.properties` file is used with afterglow as shown in Snapshot-4.4.



```
nitin@nitin:~$ cat my.csv | ./afterglow.pl -c color.properties -e 2 > my.dot
nitin@nitin:~$
```

Snapshot -4.4 : Converting CSV file into DOT file with configuration settings

Configuration file of Afterglow is `color.properties` file , which is configured as shown for this case –

```
color.source="yellow" if ($fields[0]=~/^192\.168\..*/);
color.source="greenyellow" if ($fields[0]=~/^10\..*/);
color.source="lightyellow4" if($fields[0]=~/^172\.16\..*/);
color.source="red"
color.event="yellow" if ($fields[1]=~/^192\.168\..*/)
color.event="greenyellow" if ($fields[1]=~/^10\..*/)
color.event="lightyellow4" if ($fields[1]=~/^172\.16\..*/)
color.event="red"
color.target="blue" if ($fields[2]<1024)
color.target="lightblue"
```

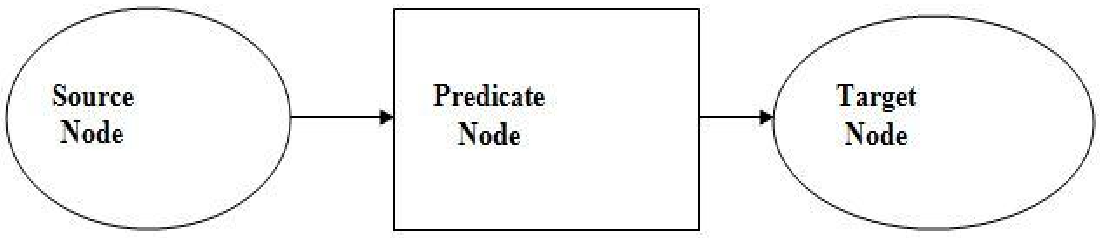
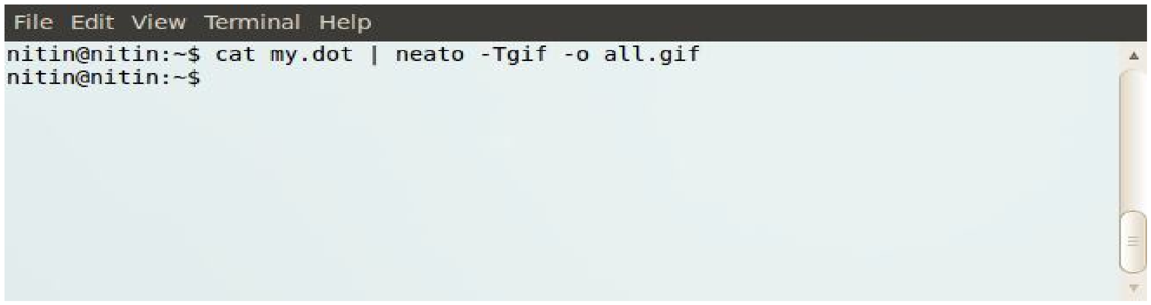


Figure 4.3 : Mapping of nodes

The property file basically consists of three assignments: `color.source`, `color.event`, and `color.target`. These values map to the three nodes as source node, predicate node , target node as shown by Figure -4.3 .

Once DOT file is generated , now data can be visualized by any of the Grapher available, which understands DOT file. GraphViz [19] provides many tools which can make graphs according to different layouts. `neato` is among one of the tools of GraphViz which generates graphs on spring model layout.



```
nitin@nitin:~$ cat my.dot | neato -Tgif -o all.gif
nitin@nitin:~$
```

Snapshot – 4.5 : Creating graph from DOT file

As shown by Snapshot ,`my.dot` file is now converted into `all.gif` file by using `neato` tool. In fact, other image file formats can also be created by `neato.`

Network traffic visualization by `all.gif` is shown in Figure – 4.4 . This is the whole traffic visualization of the network captured by `eth1.dump` . All the traffic which is coming in and going out from `eth1` interface is visually presented by afterglow [10].

In the graph shown IPs of the series 172.31.9.x are the IPs that are local to the network but are also connected to internet. 172.31.9.255 is the gateway to all the series of this IP range .That is why all the traffic of this series is passing through from the IP of gateway.

All the IPs of the series 192.168.x.x (private IPs) are shown in yellow color ,as they are defined in properties file this way .Source nodes and event nodes are shown in red.

This graph is very cumbersome to analyze since huge amount of data is being visualized. Some of the IPs are indecipherable because too many connections are crossing them and it is very hard to see those IP communication. Therefore, there is a capital need of a framework ,which can make visualization of high dimensional data easier.

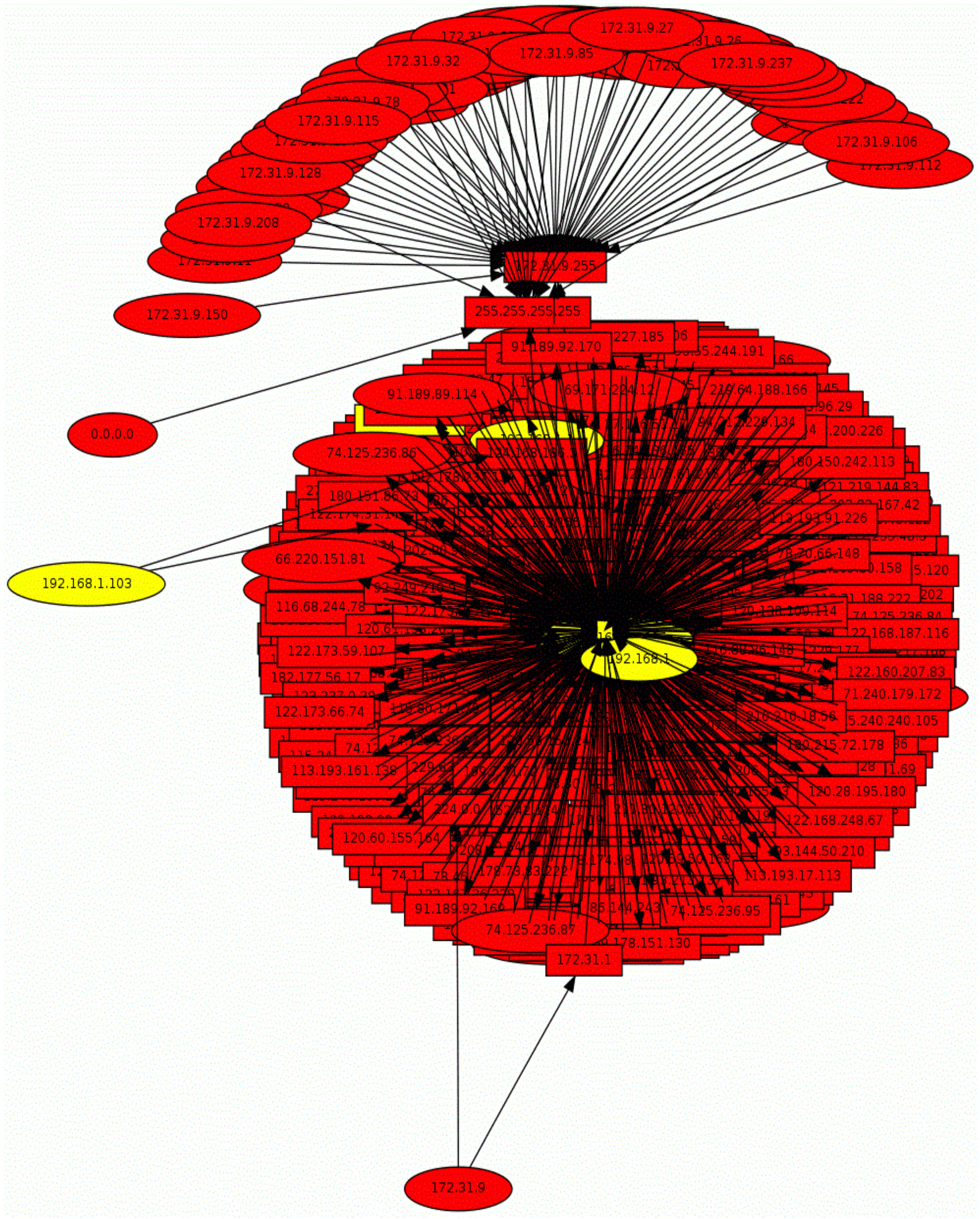


Figure 4.4 : Whole traffic recorded by the PCAP file is visualized

4.2 Development of visualization framework for university campus

The whole campus of Thapar University is connected to a centralized Network Operation Centre through which the distributions to another small local area networks are passed. This campus has thousands of nodes connected through other interconnecting devices. To ensure the security and confidentiality of data, Cyberoam provides control for identity based internet access.

4.2.1 A brief introduction to Cyberoam

Cyberoam is a Unified Threat Management System , which helps to create user identity based policies and take complete control over university' network . Cyberoam helps to assign a specific amount of bandwidth to a particular group of users. It provides extensive logging capabilities for traffic, system, and network protection functions. For Maintaining the security from unwanted traffic and other security threats, it regularly generates traffic logs. Detailed log information and reports provide historical as well as current analysis of network activity to help identify security issues and reduce network misuse and abuse. Cyberoam security tools like Firewalls, VPNs, and Proxy Servers generate a huge quantity of traffic logs which can generate a wealth of security aggregated information reports.

Cyberoam Endpoint Data Protection delivers enhanced security to organizations with policy-driven data protection and asset management over endpoints. With a full set of modules including data protection and encryption, device management, application control and asset management, it prevents data loss and supports organization-wide efforts towards security, employee productivity and efficiency in management of IT assets.

Cyberoam Central Console (CCC) with its centralized management and control offers coordinated defense against zero-hour and blended threats across distributed networks. It enables enterprise-wide implementation of corporate Internet policy, ensuring high productivity and security.

Cyberoam UTM appliances integrate multiple security features like Firewall, VPN, Intrusion Prevention System, Anti-Virus & Anti-Spyware, Anti-Spam, Web Filtering, Layer 7 Visibility & Control, Bandwidth Management, Multiple Link Management, and more on a single platform. Extensible Security Architecture along with multi core processors enable it to offer future-ready security and faster throughput.

Cyberoam iView offers visibility into activity within the organization for high levels of security, data confidentiality & regulatory compliance. It provides an organization-wide security picture on a single dashboard through centralized reporting of multiple devices across geographical locations.

4.2.2 Proposed Framework

To analyze these voluminous logs is very awkward task ,since they produce plethora of data which cannot be easily examined. Moreover , on creating graphs of such data is still problematic because thousands of nodes cannot be easily shown on a graph. So an efficient framework is needed for Thapar University for visualization sizeable amount of data. Therefore, a visualization framework is proposed and further tested for a large and diverse set of data logs , based on filtering and reducing the data to discover the concerned data .Figure -4.5 shows the framework -

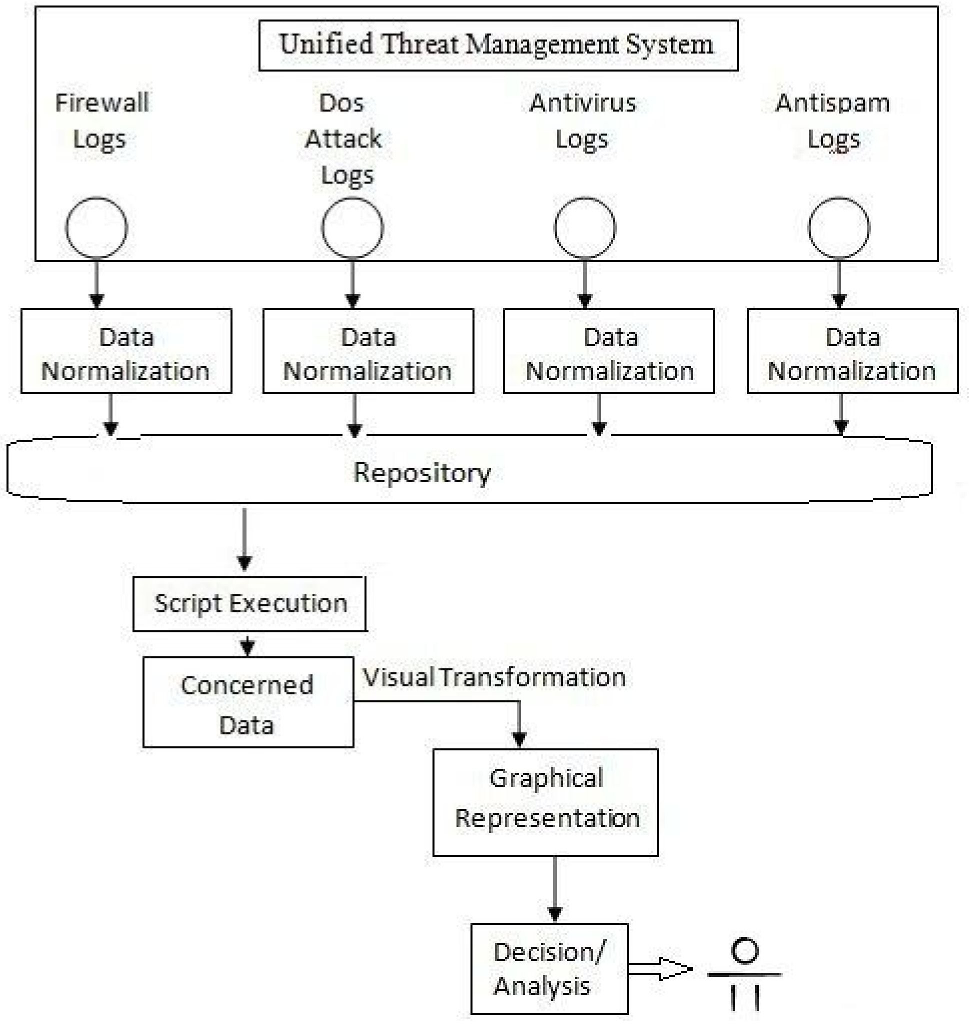


Figure 4.5 : Proposed framework of Visualization

This framework has four main components which play the vital role in reducing the data and perform efficient data visualization –

i- Data Normalization- The data we collected through Unified Threat Management System is high dimensional data incorporating various logs. These logs can be firewall logs, anti spam logs, antivirus logs etc. The data of these logs is not in the right format that is required to input for visualization application. This component first processes the data and commute the data in the needed format. Generally , the aim of data normalization is to convert the data captured from the logs in to comma separated values format.

Thus all the useful fields are separated by commas and thus normalized data is saved into the repository ,where data from different sources is stored. All the comma separated fields are stored here and according to need of visualization they are chosen. Whenever any particular field is required to be shown in graph, it is fetched form the repository and further used.

ii- Script Execution- It is the most important component in the framework which is used to reduce the data and filter the unnecessary data . Corresponding normalized log data is fetched from the repository .Then script is executed for filtering the unnecessary data. After Script Execution we get only concerned data that is of our interest. This script reduces the cognitive burden of administrator by reducing the inherent data. Concerned data is the filtered useful data.

iii- Graphical Representation- The concerned data is converted to graphical form through visual transformation. This is the general approach for creating graphs as discussed in earlier section in which it is tried to use graph language file and by the help of grapher graphs are created.

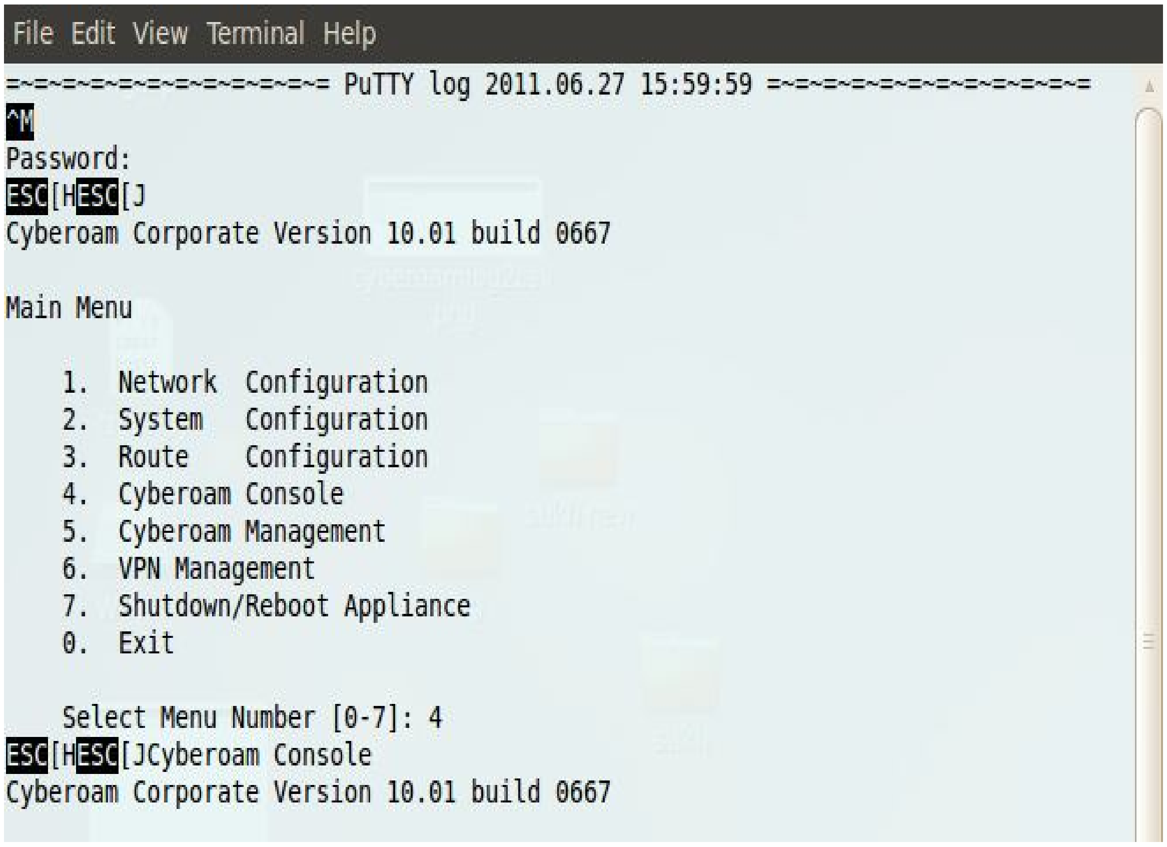
iv- Decision and Analysis Module- By visual perception of graph we can analyze and take decision which host is producing malicious activities. There are some patterns by which we can analyze that a particular host is malicious. Consider the case of ping sweep, where intruder tries to send simultaneous ping request to different hosts. By examining the graphs for such kind of patterns ,malicious activities going on in the campus can be analyzed.

4.3 Demonstration of the framework in large scale university network

To verify the significance of the above framework ,this is deployed in large scale network of Thapar University. Following are the steps performed to demonstrate the use of this framework-

4.3.1 Capturing Cyberoam log data

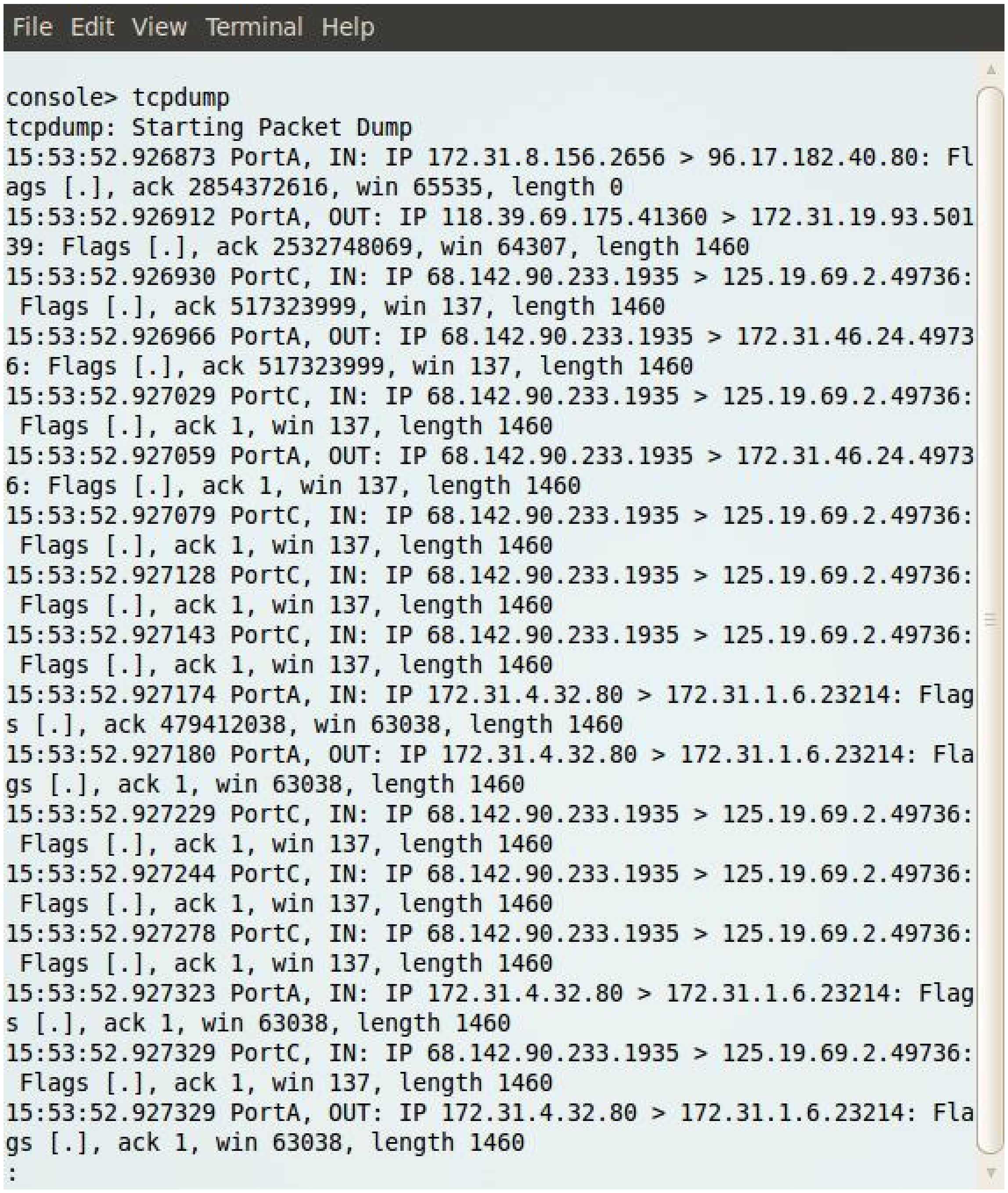
Log data is collected from Cyberoam which is a unified threat management system. The vastness of the network is imagined such a way that capturing log file for 60 sec created data of 137 MB. It shows two things ,the immensity of university network that thousands of nodes are live on network and another is that there is need of efficient data visualization because to analyze file of such size is not a good idea. Snapshot -4.6 shows the snippet of the management window of Cyberoam .



```
=~=~=~=~=~=~=~=~=~=~=~= PuTTY log 2011.06.27 15:59:59 =~=~=~=~=~=~=~=~=~=~=~=
^M
Password:
ESC[HESC[J
Cyberoam Corporate Version 10.01 build 0667

Main Menu

    1.  Network  Configuration
    2.  System   Configuration
    3.  Route    Configuration
    4.  Cyberoam Console
    5.  Cyberoam Management
    6.  VPN Management
    7.  Shutdown/Reboot Appliance
    0.  Exit

    Select Menu Number [0-7]: 4
ESC[HESC[JCyberoam Console
Cyberoam Corporate Version 10.01 build 0667
```

Snapshot – 4.6 : Cyberoam Management

Snapshot -4.7 shows the Cyberoam console under Cyberoam management. Currently this Snapshot showing the packet capturing process run by Cyberoam prompt.



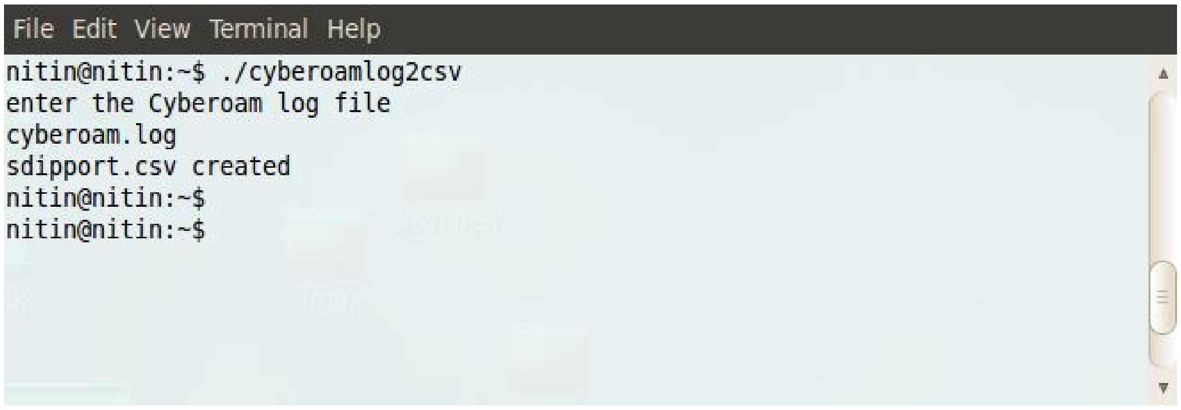
```
console> tcpdump
tcpdump: Starting Packet Dump
15:53:52.926873 PortA, IN: IP 172.31.8.156.2656 > 96.17.182.40.80: Fl
ags [.], ack 2854372616, win 65535, length 0
15:53:52.926912 PortA, OUT: IP 118.39.69.175.41360 > 172.31.19.93.501
39: Flags [.], ack 2532748069, win 64307, length 1460
15:53:52.926930 PortC, IN: IP 68.142.90.233.1935 > 125.19.69.2.49736:
 Flags [.], ack 517323999, win 137, length 1460
15:53:52.926966 PortA, OUT: IP 68.142.90.233.1935 > 172.31.46.24.4973
6: Flags [.], ack 517323999, win 137, length 1460
15:53:52.927029 PortC, IN: IP 68.142.90.233.1935 > 125.19.69.2.49736:
 Flags [.], ack 1, win 137, length 1460
15:53:52.927059 PortA, OUT: IP 68.142.90.233.1935 > 172.31.46.24.4973
6: Flags [.], ack 1, win 137, length 1460
15:53:52.927079 PortC, IN: IP 68.142.90.233.1935 > 125.19.69.2.49736:
 Flags [.], ack 1, win 137, length 1460
15:53:52.927128 PortC, IN: IP 68.142.90.233.1935 > 125.19.69.2.49736:
 Flags [.], ack 1, win 137, length 1460
15:53:52.927143 PortC, IN: IP 68.142.90.233.1935 > 125.19.69.2.49736:
 Flags [.], ack 1, win 137, length 1460
15:53:52.927174 PortA, IN: IP 172.31.4.32.80 > 172.31.1.6.23214: Flag
s [.], ack 479412038, win 63038, length 1460
15:53:52.927180 PortA, OUT: IP 172.31.4.32.80 > 172.31.1.6.23214: Fla
gs [.], ack 1, win 63038, length 1460
15:53:52.927229 PortC, IN: IP 68.142.90.233.1935 > 125.19.69.2.49736:
 Flags [.], ack 1, win 137, length 1460
15:53:52.927244 PortC, IN: IP 68.142.90.233.1935 > 125.19.69.2.49736:
 Flags [.], ack 1, win 137, length 1460
15:53:52.927278 PortC, IN: IP 68.142.90.233.1935 > 125.19.69.2.49736:
 Flags [.], ack 1, win 137, length 1460
15:53:52.927323 PortA, IN: IP 172.31.4.32.80 > 172.31.1.6.23214: Flag
s [.], ack 1, win 63038, length 1460
15:53:52.927329 PortC, IN: IP 68.142.90.233.1935 > 125.19.69.2.49736:
 Flags [.], ack 1, win 137, length 1460
15:53:52.927329 PortA, OUT: IP 172.31.4.32.80 > 172.31.1.6.23214: Fla
gs [.], ack 1, win 63038, length 1460
:
```

Snapshot 4.7 : Capturing packets through Cyberoam console

4.3.2 Creation of script for conversion of Cyberoam log data into CSV

Through Cyberoam management console ,the log data is collected and saved as `cyberoam.log.` To normalize the Cyberoam data , there is need to use a script which can convert this data into appropriate format required for visualization. Therefore, a script is created which can convert this log data into comma separated values. It is required to change this log data and make it the way that all the source IP , destination IP, source port and destination port are separated by commas. Further they can be used for specific role in network visualization.

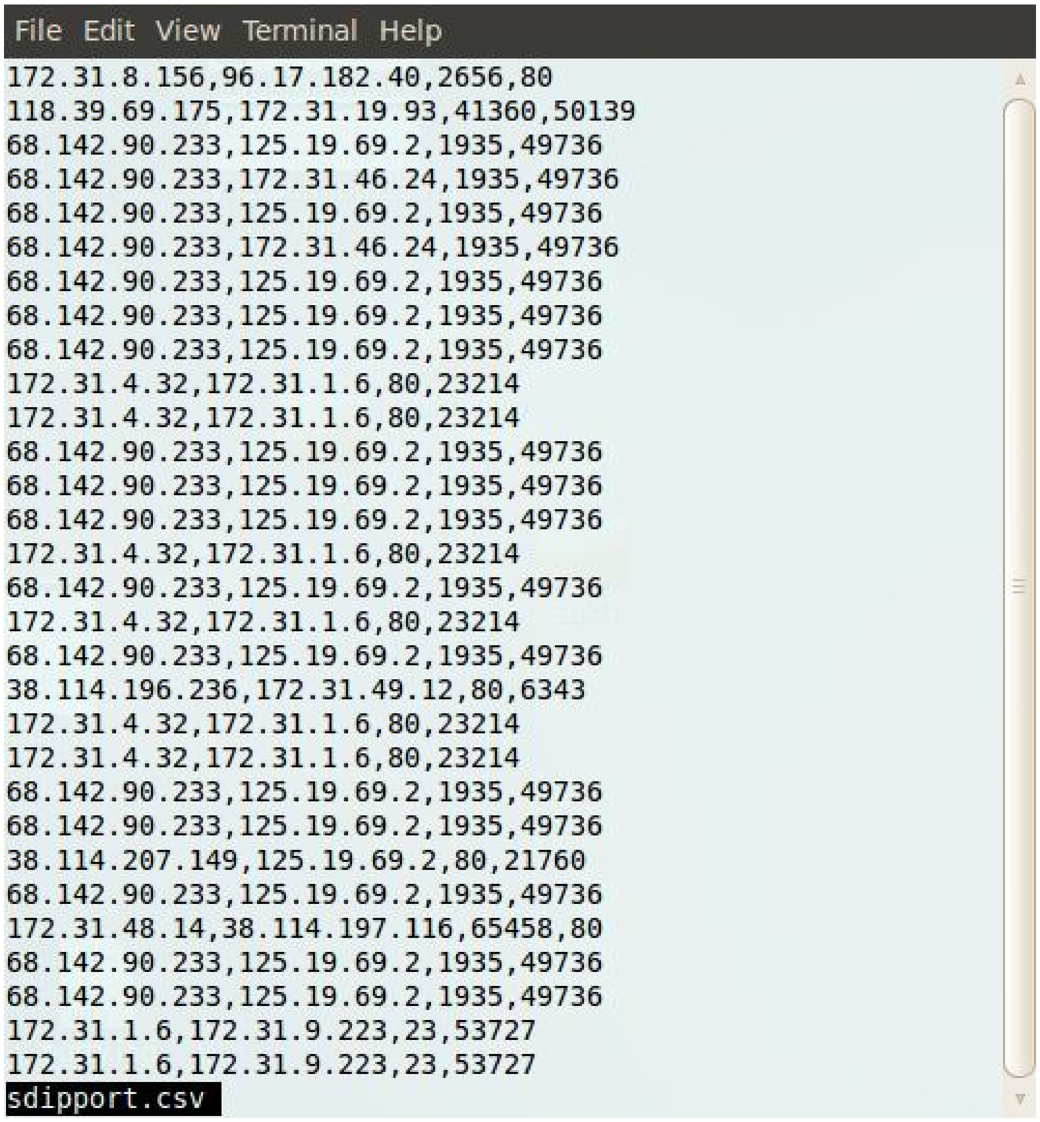
`Cyberoamlog2csv` is the script that is created to convert `cyberoam.log` into `sdipport.csv`. Snapshot -4.8 shows conversion into `sdipport.csv`.



```
nitin@nitin:~$ ./cyberoamlog2csv
enter the Cyberoam log file
cyberoam.log
sdipport.csv created
nitin@nitin:~$
nitin@nitin:~$
```

Snapshot 4.8 : Converting cyberoam.log into sdipport.csv

`sdipport.csv` is created , which looks like as shown by its snippet in Snapshot -4.9. Among all the fields captured in log file source IPs ,destination IPs , source port number , destination port numbers are extracted and are put together separated by commas .



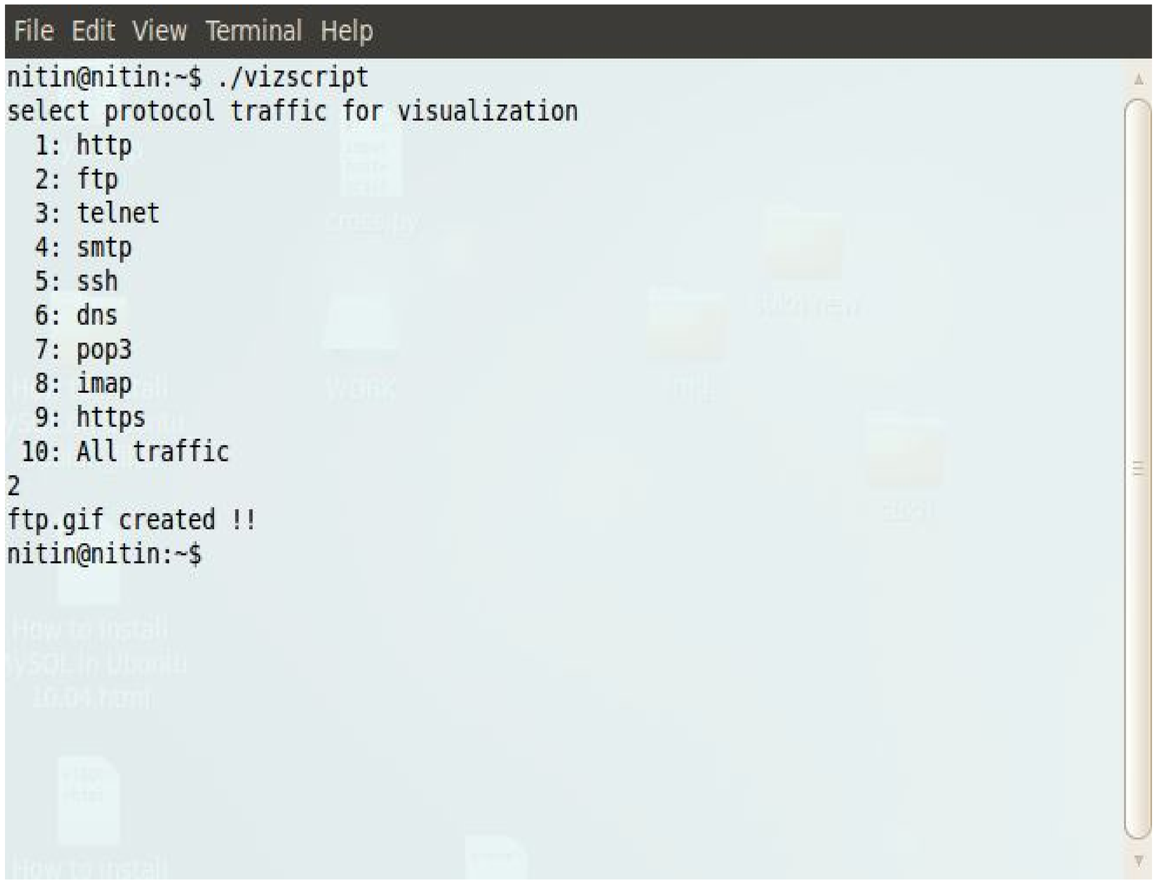
```
172.31.8.156,96.17.182.40,2656,80
118.39.69.175,172.31.19.93,41360,50139
68.142.90.233,125.19.69.2,1935,49736
68.142.90.233,172.31.46.24,1935,49736
68.142.90.233,125.19.69.2,1935,49736
68.142.90.233,172.31.46.24,1935,49736
68.142.90.233,125.19.69.2,1935,49736
68.142.90.233,125.19.69.2,1935,49736
68.142.90.233,125.19.69.2,1935,49736
172.31.4.32,172.31.1.6,80,23214
172.31.4.32,172.31.1.6,80,23214
68.142.90.233,125.19.69.2,1935,49736
68.142.90.233,125.19.69.2,1935,49736
68.142.90.233,125.19.69.2,1935,49736
172.31.4.32,172.31.1.6,80,23214
68.142.90.233,125.19.69.2,1935,49736
172.31.4.32,172.31.1.6,80,23214
68.142.90.233,125.19.69.2,1935,49736
38.114.196.236,172.31.49.12,80,6343
172.31.4.32,172.31.1.6,80,23214
172.31.4.32,172.31.1.6,80,23214
68.142.90.233,125.19.69.2,1935,49736
68.142.90.233,125.19.69.2,1935,49736
38.114.207.149,125.19.69.2,80,21760
68.142.90.233,125.19.69.2,1935,49736
172.31.48.14,38.114.197.116,65458,80
68.142.90.233,125.19.69.2,1935,49736
68.142.90.233,125.19.69.2,1935,49736
172.31.1.6,172.31.9.223,23,53727
172.31.1.6,172.31.9.223,23,53727
sdipport.csv
```

Snapshot 4.9 : A snippet of sdipport.csv

4.3.3 Creation and execution of script for visualization framework

Now there is need to create a script which can filter the unnecessary data and can provide only the useful data for visualization. This is the concerned data that is only required in graph creation. Together, there is also a need of a script which can visually represent the data in the form of graphs.

Therefore, a script named `vizscript` is created for fulfilling the purpose of framework of visualization. This script not only used for filtering and reducing the data but also used for visual transformation of that data. The function and working of the script is shown in Snapshot-4.10 .



```
nitin@nitin:~$ ./vizscript
select protocol traffic for visualization
  1: http
  2: ftp
  3: telnet
  4: smtp
  5: ssh
  6: dns
  7: pop3
  8: imap
  9: https
 10: All traffic
2
ftp.gif created !!
nitin@nitin:~$
```

Snapshot 4.10: Running vizscript for getting FTP traffic

Thus the script works for filtering the data on the basis of protocol. By this way only data of a particular protocol is visualized and further it can be easily analyzed. Without using this script it is not possible to analyze the graphs because user feels himself unable to differentiate among data which is coming in and passing out from different IPs. Once user has understanding that only data of a particular protocol is visualized, it lets user analyze data very easily.

In the case of Cyberoam where we captured log data as `cyberoam.log` , on applying this proposed framework we get graphs , that are easier to analyze. On selecting the traffic of protocol FTP , a graph showing FTP sessions is created as `ftp.gif` as shown in Figure-4.6.

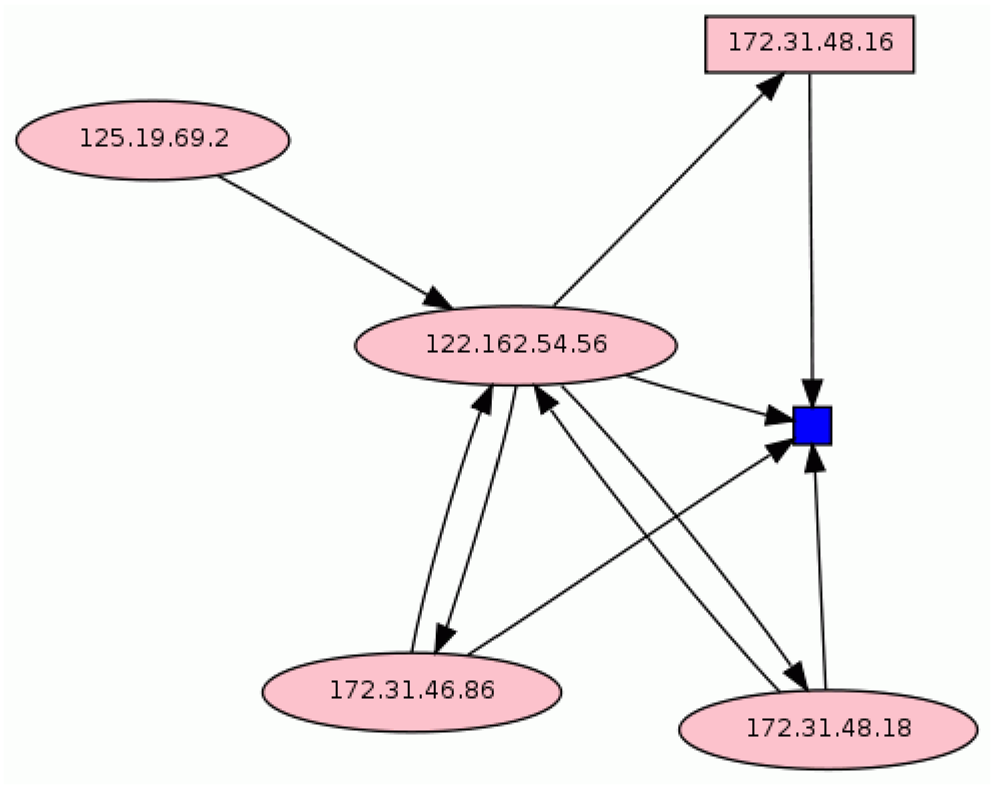
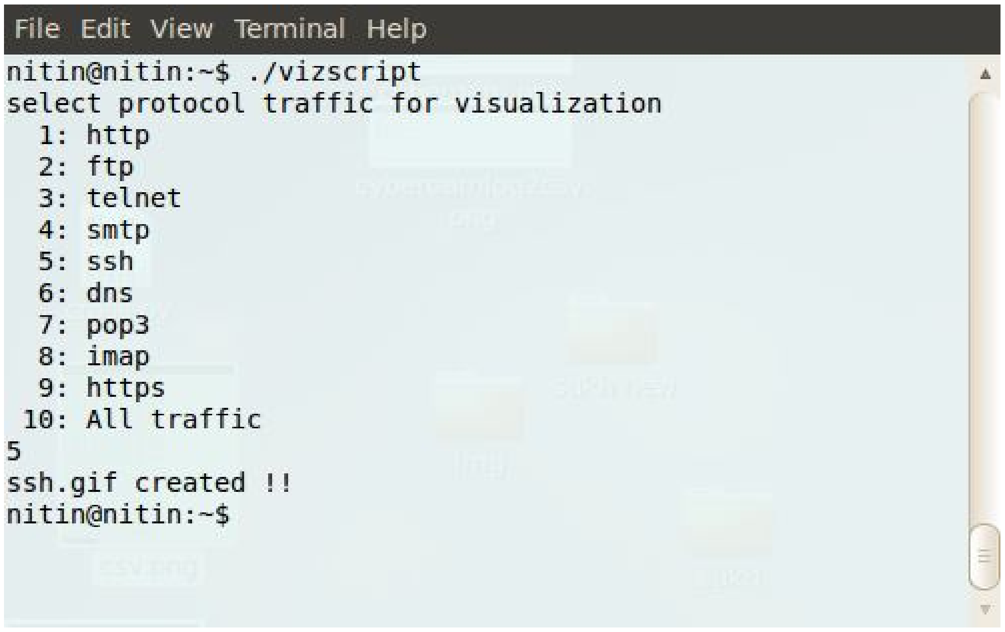


Figure 4.6 : FTP traffic after applying the framework

At the time , when `cyberoam.log` is captured , only PCs with these IPs were performing the FTP sessions. IP 172.31.48.16 is the event node , which is acting as a predicate node. Whereas other IPs are source and destination IPs and are indulged in FTP sessions through IP 172.31.48.16.

Similarly for visualizing the SSH protocol traffic , we performed-



```
nitin@nitin:~$ ./vizscript
select protocol traffic for visualization
  1: http
  2: ftp
  3: telnet
  4: smtp
  5: ssh
  6: dns
  7: pop3
  8: imap
  9: https
 10: All traffic
5
ssh.gif created !!
nitin@nitin:~$
```

Snapshot – 4.11 : Running vizscript for getting SSH traffic

As the output `ssh.gif` generated, which is shown in Figure – 4.7. To obtain the visualization of SSH traffic.

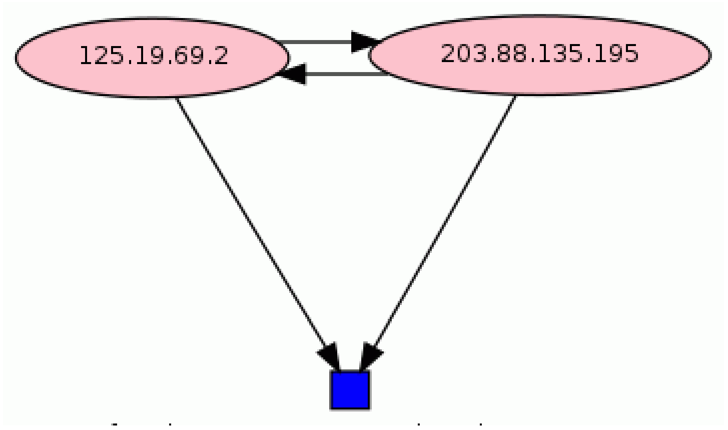


Figure 4.7 : SSH traffic after applying the framework

As it can be seen in figure- 4.7 ,only two hosts with IP 125.19.69.2 and 203.88.135.195 are indulged in SSH session , and it can be easily visualized that there is no malicious host.

Similar actions are performed and `SMTP.gif` is obtained as shown in Figure – 4.8.

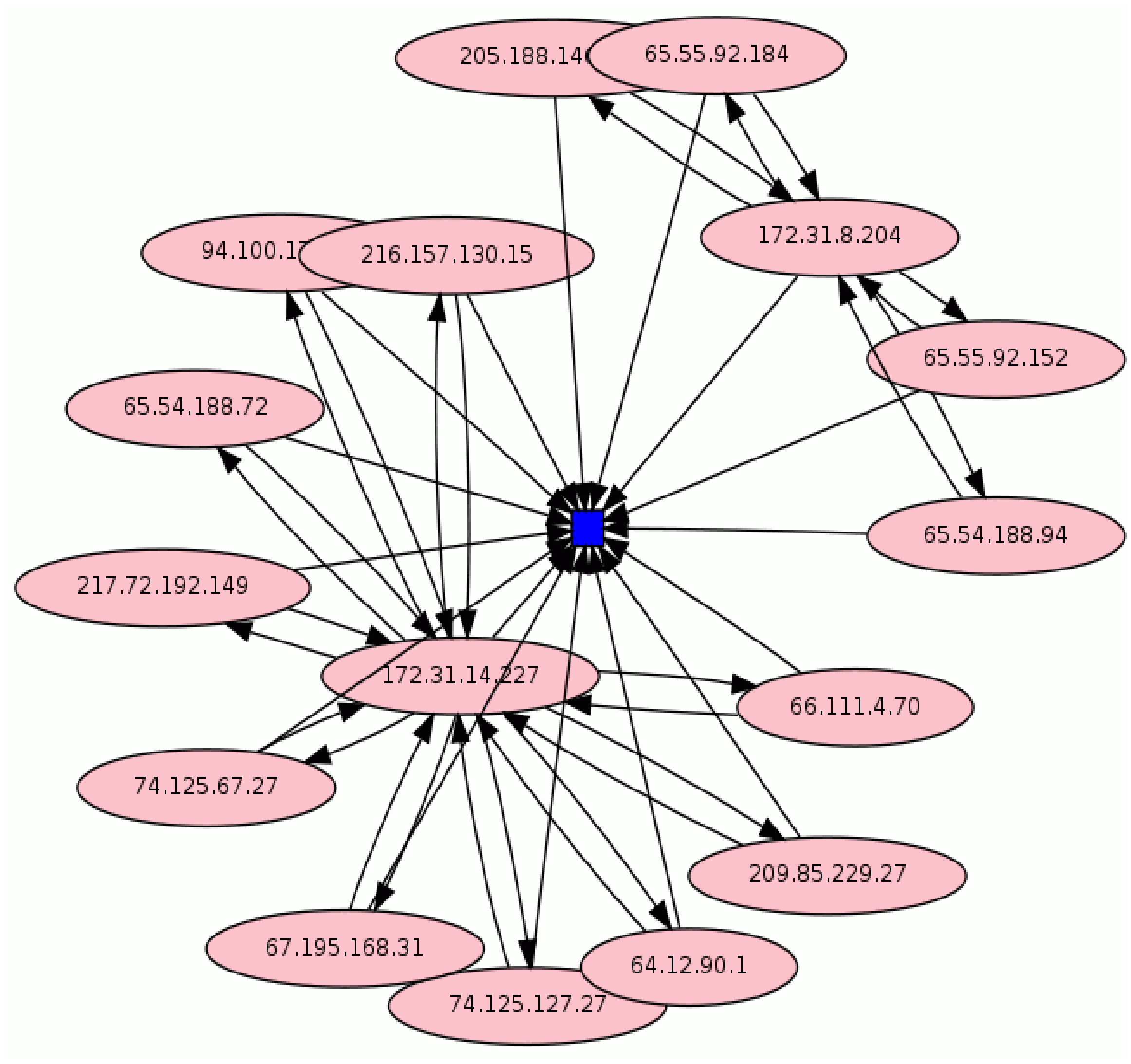


Figure 4.8 : SMTP traffic after applying the framework

As seen by Figure-4.8 while visualizing SMTP traffic two IPs 172.31.14.227 and 172.31.8.204 are the IPs of Thapar University. They both are trying to access the mail servers as they both make incoming edges to the mail server. Therefore, these two IPs in the university network are malicious.

4.4 Problems faced during Implementation

- **Missing Text/CSV.pm-** While parsing CSV file through perl ,missing CSV.pm problem was faced . CSV.pm is included in the package @INC. So the package @INC is downloaded from CPAN library and installed.

- **Need of specific version of tcpdump -** Initially tcpdump version 4.1.1 was installed on the ubuntu box. While using the script tcpdump2csv.pl , it started showing peculiar behavior. Further another version of tcpdump 3.9.8 is installed ,then only the expected output is achieved.

- **Irregular data format in different tcpdump versions-** Different versions of tcpdump captures and displays the data in different formats. While collecting Cyberoam log data even from tcpdump ,the display format was quite different. So another script is being written to get the required data.

Chapter 5

Conclusion and Future scope

5.1 Conclusion

Increasing use of computer network technologies has brought the world together. The security technologies that help to remain away these vast networks from malicious activities have not been progressed as rapidly. Network visualization is the nascent field which can help administrator to take security decisions quickly. Though, Network traffic visualization is not the panacea of all the security problems, but if used the right way in combination with other network security tools , it can be beneficial.

This thesis explored and investigated the various visualization approaches and methodologies. The main results are as follows :

- Design and developed the enhanced framework of visualization for Thapar University. This framework tries to reduce and filter the unnecessary traffic data for visualization and reduce the cognitive burden of administrator.
- Demonstrated the usefulness of proposed network for large network of Thapar University .This framework captures the huge traffic data from Cyberoam log files and visualize the network by percolating the data through scripts execution.

5.2 Future Scope

This work can be extended in different directions :

- Automation of the whole framework can be done. Triggers can be generated whenever any malicious host is found.
- Reduction in time consumed while graphs of the very huge data are prepared.

References

[1] Raffael Marty, “ Applied Security Visualization ”, Addison Wesley Professional , ISBN-13: 978-0-321-51010-5 .

[2] James J. Thomas and Kristin A. Cook eds , “ Illuminating the Path –The Research and Development Agenda for Visual Analytics”, IEEE Computer Society Press, 2005.

[3] Joe Burrescia and William Johnston , ESnet Status Update , Internet 2 International Meeting, 2005.

[4] E. Wes Bethel ,Scott Campbell ,Eli Dart, “Accelerating Network Traffic Analytics Using Query-Driven Visualization” , Visual Analytics Science And Technology, 2006 IEEE Symposium .

[5] Greg Conti , “Security Data Visualization: Graphical Techniques for Network Analysis” ,No Starch Press, September 2007.

[6] Colin Ware, “Information visualization perception for design”, Morgan Kaufmann , 2004.

[7] Roberto Tamassia1, Bernardo Palazzi1, and Charalampos Papamanthou1, “Graph Drawing for Security Visualization”, Springer-Verlag Berlin, Heidelberg ,2009

[8] H. Debar, D. Curry, B. Feinstein , “ The Intrusion Detection Message Exchange Format (IDMEF)” , Request for Comments: 4765 ,March 2007 .

[9] Common Event Expression :The Log Standard , http://cee.mitre.org/

[10] Libpcap : Packet capturing and filtering , http://sourceforge.net/projects/libpcap/

[11] Winpcap : A tool for link-layer network access in Windows environments , http://www.winpcap.org/

[12] Wireshark : An open source packet analyzer ,http://www.wireshark.org/

[13] Tcpdump :A command line open source packet analyzer, http://www.tcpdump.org

[14] Yuri Engelhardt and Juan C. Dursteler ,The Infovis Diagram , http://www.infovis.net/printMag.php?num=187&lang=2

[15] Erwan Le Mal ´ecot, Masayoshi Kohara, Yoshiaki Hori, Kouichi Sakurai, “Interactively Combining 2D and 3D Visualization for Network Traffic Monitoring” , VizSEC '06 : Proceedings of the 3rd international workshop on Visualization for computer security.

[16] D. A. Keim, “Information visualization and visual data mining” ,IEEE Transactions on Visualization and Computer Graphics, 8(1), 1-8, 2002.

[17] M. Allen, P. McLachlan, “NAV Network Analysis Visualization” , Computer Modeling and Simulation, 2009, EMS '09.

[18] Shafranovich, Y. , Common Format and MIME Type for Comma-Separated Values (CSV) Files , RFC 4180, The Internet Society .

[19] Graphviz , a AT&T Labs Research , http://graphviz.org/

[20] Dot language : A plain text graph description language , http://www.graphviz.org/content/dot-language

[21] Graph Modelling Language : A hierarchical ASCII-based file format for describing graphs, http://en.wikipedia.org/wiki/Graph_Modelling_Language

[22] Douglas R. Mauro and Kevin J. Schmidt, “Essential SNMP”, O'Reilly & Associates, 2001.

[23] D. Perkins, E. McGinnis, “Understanding SNMP MIBS”, Prentice Hall, 1996.

[24] R. Ball, G. A. Fink, and C. North, “Home-centric visualization of network traffic for security administration” , VizSEC/DMSEC ’04:Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security, pages 55–64. ACM Press, 2004.

[25] Jonathan McPherson, Kwan-Liu Ma, Paul Krystosk, Tony Bartoletti,and Marvin Christensen, “Portvis: a tool for port-based detection of security events” ,VizSEC/DMSEC ’04: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security, pages 73–81. ACM Press, 2004.

[26] Kiran Lakkaraju, William Yurcik, and Adam J. Lee. “NVisionIP: netflow visualizations of system state for security situational awareness.” ,VizSEC/DMSEC ’04: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security, pages 65–72.

[27] Oetiker, Tobias and Rand, Dave, “Multi Router Traffic Grapher.”, Version 2.16.4 , May 17, 2010.

[28] Oetiker, Tobias and Rand, Dave. “Round Robin Database Tool.” , Version 1.4.4. July 5, 2010.

[29] Raffael Marty , http://afterglow.sourceforge.net/.

[30] Doris Wong Hooi Ten and Sureswaran Ramadass, “Case Study: Visualization Methodology For Analysing Network Data” ,March 2010 12th International Conference on Computer Modelling and Simulation .

[31] D. S. C. Russo, P. Gros, P. Abel, D. Loisel, J-P Paris, “Using Virtual Reality for Network Management: Automated Construction of Dynamic 3D Metaphoric Worlds,” Symposium on Virtual Reality Software and Technology, 1999 .

[32] Doris Hooi-Ten Wong, Kok-Soon Chai, Sureswaran Ramadass, Nicolas Vavasseur, “Expert-Aware Approach:A New Approach To Improve Network Security Visualization Tool”, July 2010 Second International Conference on Computational Intelligence, Communication Systems and Networks.

[33] Zhang Jiawan , Li Liang , Lu Liangfu1 , Zhou Ning , “A Novel Visualization Approach for Efficient Network Scans Detection” , International Conference on Security Technology , 2008 .

Paper Publication

Nitin Prakash Verma, Maninder Singh, “A hybrid and enhanced Framework for Visualization and Analysis of Network Traffic Data” , Paper is communicated to Research Journal of Computer Systems Engineering- RJCSE- an International Journal (ISSN: 2230-8563; e-ISSN: 2230-8571)