# Information Extraction for Offline Traffic Anomaly Detection in NIDS

**Renuka Prasad.B, Dr.Annamma Abraham, Chandan. C, Prabhanjan.A, AjayBilotia**
Department of MCA, R.V.College of Engineering, Bangalore, Karnataka,India 560-059

**Summary**
This paper discusses an efficient approach for extracting information from the libpcap compatible network data using scripting languages. The extracted information will be used for traffic anomaly detection in network intrusion detection systems. The information extraction is done on DARPA 1998 dataset. Extraction process includes exporting the dataset into plain text or csv format using wireshark (network sniffing tools) and then extracting the information using scripting languages. The results obtained from the scripts written using gawk, awk, sed and perl are compared with the results obtained from the DDL, DML, procedures given to the database. It is found that the use of scripts along with queries was more efficient than using just using DDL, DML, procedures on database or scripting languages alone for extracting information

***Keywords:***
*IDS, NIDS, anomaly detection, information extraction,gawk, DDL, DML, sed, pager.*

## 1. Introduction

Computer or Computer network security can be classified into three main modules such as prevention, detection and response / protection**[1]**. Prevention modules comprises of techniques which prevent any security breaches. Prevention modules include precautionary measures such as cryptographycal authentication and authorization techniques which prevent any attacks or intrusions before they happen. Detection modules are one step ahead, in that it expects security breaches to happen and tries to detect them. Alerts are generated and sent to the system administrator if a known attack is detected. It can also predict and forecast any new attacks. The Protection modules comprises of actions or measures taken by an administrator for protecting a computer or computer network.

In this paper we discuss about IDS, types of IDS. variety of attacks and some of the attack detection techniques. Also discussed is about the format that is required for statistical traffic anomaly detection and efficient information extraction techniques from a libpcap compatible network data.

Manuscript received September 5, 2008.
Manuscript revised September 20, 2008.

## 2. Intrusion Detection Systems ( IDS )

**2.1 IDS [3]** monitors a computer or a computer network always and generates an alert to the administrator in case of any security breaches like anomalous activities of an user or any malicious activity by a user or an external system. Recent developments in IDS does include actions been taken over an attack. Intrusion Detection Systems (IDS) are generally classified into two types called Host Intrusion Detection Systems (HIDS) and Network Intrusion Detection Systems (NIDS) also a variation of these called Hybrid Intrusion Detection Systems also exist.

### 2.2 HIDS

**HIDS[4]** monitors a single host ( computer ) for any attacks or intrusions that might happen on it and generates alerts accordingly

### 2.3 NIDS

**NIDS[4]** monitors a specified computer network for any attack that happens from inside the network and also from outside the network.

## 3. Attacks

Computer system attack classification is done very well by N. Paulauskas, E. Garsva **[2]** and the general strategies for intrusion detection include signature verification, bottleneck verification, specification based and anomaly detection approaches

A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems by Kendall **[7]** has given a taxonomy of computer attacks in a simpler way so as to make the task of Intrusion detection system to evaluate DARPA98 **[6]** easy.

### 3.1. Classification of Attacks

In our paper we consider the datasets with attacks given by MIT for evaluation of DARPA 1998 dataset competition. **[5]**

i). **Denial of Service Attacks** (Apache2, arppoison, Back, Crashiis, dosnuke, Land, Mailbomb , SYN Flood , Ping of Death,
ProcessTable,selfping,Smurf,sshprocesstable,Syslogd,tcpr

eset,Teardrop,Udpstorm)
ii).**User to Root Attacks** (anypw, casesen, Eject, fbconfig, Fdformat, Loadmodule, ntfsdos, Perl, Ps, sechole, Xterm, yaga.)
iii). **Remote to Local Attacks** (Dictionary, Ftpwrite, Guest, Httptunnel, Imap, Named, ncftp, netbus, netcat, Phf, ppmacro, Sendmail, sshtrojan , Xlock, Xsnoop)
iv).**Probes** (insidesniffer, Ipsweep, ls_ domain, Mscan, NTinfoscan, Nmap, queso, resetscan, Saint, Satan)
v). **Data** (Secret)
Denial of Service attacks, User to Root, Remote to Local and Data Attacks can be classified as attacks by objective, Probe attacks can be classified as attacks by effect **[2].**

## 3.2. Detection Techniques

Some of the main approaches for detecting intrusions are discussed in **[2]** include signature verification, bottleneck verification, specification based detection and anomaly based detection

### 3.2.1. Signature Based Detection

Signature based detection needs very less memory and CPU time as it can only detect those attack which are already know to the IDS as it is just comparing any unusual or doubtful behavior (sequence of events) of a user or a system or a program over a host or on a network with the predefined set of signatures, this approach is effective only for known attacks and fails when attacks are made.

### 3.2.2 Bottle Neck Verification

Bottle neck verification approaches are useful to identify user to root attacks where the states of the users are very well defined and are monitored regularly this approach needs lot of CPU time and memory. Few new attacks can be identified by this approach.

### 3.2.3. Specification Based

Specification based approaches are for detecting intrusions are good for detecting any violations of the specifications defined as normal behaviors of programs that is improper use of the system or application programs. This approach detects new attacks but requires more time and memory than bottleneck verification techniques.

### 3.2.4 Anomaly Based Detection

Anomaly based detection can detect traffic anomalies, protocol anomalies and data anomalies. This is the best approach which could detect any new type of attacks. Statistical models are used to identify the deviations from the normal behavior of the traffic, protocol or data and generate an alert to the administrator. Anomaly based detection techniques requires lot of memory and CPU time.

we focus on efficient information extraction from the network data for statistical based anomaly detection for detecting new attacks. In anomaly detection, the system administrator will define the thresholds both at minimum and maximum, also defines normal network's traffic load, breakdown, protocol, and typical packet size. The anomaly detector monitors networks to compare their state to the normal conditions and observes deviations and generates alerts for any anomalies when they occur. The administrator will have to train the Security system continuously about the thresholds and see that the maximum attacks are detected and alert rates are reduced

## 4. Information Extracting Tools and Their Reporting Format

### 4.1 tcpdump

tcpdump is one of the oldest and widely used software to monitor the network and dump the packets for later analysis. MIT's DARPA 1998 data set was also obtained by using **tcpdump[8]**. Using tcpdump to extract information for detection any intrusions is very time consuming.
For example :

```
# tcpdump -ne dst port 80 and 'tcp[13] & 2 == 2'
This way effectively filtering only SYN packets on
port 80.
 # tcpdump -c 30000 -ne dst port 80 and 'tcp[13] & 2
== 2' | awk '{print $11}' | cut -d. -f1|sort | uniq -c |
sort -n
Dumping 30K packets,cutting the first octet from the
IPs and sorting by number of packets originating
from this A class net.
```

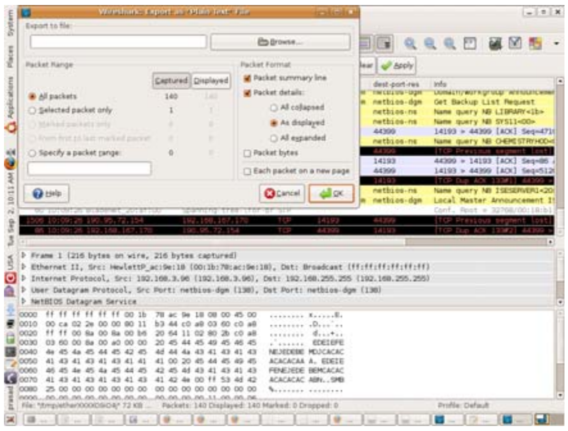
Since the data collected from tcpdump will be libpcap**[18]** format we need some other tools for extracting information form the libpcap files.

```
 Display all URG packets
# tcpdump 'tcp[13] & 32 != 0'
 Display all ACK packets
 # tcpdump 'tcp[13] & 16 != 0'
 Display all PSH packets
 # tcpdump 'tcp[13] & 8 != 0'
 Display all RST packets
 # tcpdump 'tcp[13] & 4 != 0'
  Display all SYN packets
 # tcpdump 'tcp[13] & 2 != 0'
 Display all FIN packets
 # tcpdump 'tcp[13] & 1 != 0'
 Display all SYN-ACK packets
 # tcpdump 'tcp[13] = 18'
```

From the above commands to extract some information shows that this is not enough to make any conclusions as getting the counts of different types of packets and also getting certain packets within some duration is also difficult.

**Ethereal and wireshark**

Ethereal and wireshark **[9]** allows to export the libpcap compatible files to different forms like plaint text format, one packet per page with complete details and also CSV format which will be very useful for further analysis. But again what is required is not served by ethereal or wireshark at once. The exported formats will look like plain-text space separated format or csv formatted.



Packet Range: [✓ All packets] 140; Selected packet only 1; Specify a packet range: 0

Packet Format: [✓ Packet summary line] [✓ Packet details:] All collapsed [✓ As displayed] All expanded; Packet bytes; Each packet on a new page

```
netbios-dgm   Get Backup List Request
netbios-ns    Name query NB LIBRARY<1b>
netbios-ns    Name query NB SYS11<00>
44399         14193 > 44399 [ACK] Seq=471
netbios-ns    Name query NB CHEMISTRYHOD<
44399         [TCP Previous segment lost]
14193         44399 > 14193 [ACK] Seq=86
44399         14193 > 44399 [ACK] Seq=512
14193         [TCP Dup ACK 133#1] 44399 >
netbios-ns    Name query NB ISESERVER1<20
netbios-dgm   Local Master Announcement I
1506 10:09:26 190.95.72.154   192.168.167.170  TCP  14193  44399  [TCP Previous segment lost]
  86 10:09:26 192.168.167.170 190.95.72.154    TCP  44399  14193  [TCP Dup ACK 133#2] 44399 >

Frame 1 (216 bytes on wire, 216 bytes captured)
Ethernet II, Src: HewlettP_ac:9e:18 (00:1b:78:ac:9e:18), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Internet Protocol, Src: 192.168.3.96 (192.168.3.96), Dst: 192.168.255.255 (192.168.255.255)
User Datagram Protocol, Src Port: netbios-dgm (138), Dst Port: netbios-dgm (138)
NetBIOS Datagram Service

0000  ff ff ff ff ff ff 00 1b  78 ac 9e 18 08 00 45 00   ........ x.....E.
0010  00 ca 02 2e 00 00 80 11  b3 44 c0 a8 03 60 c0 a8   ........ .D...`..
0020  ff ff 00 8a 00 8a 00 b6  20 64 11 02 80 2b c0 a8   ........  d...+..
0030  03 60 00 8a 00 a0 00 00  20 45 44 45 49 45 46 45   .`......  EDEIEFE
0040  4e 45 4a 45 44 45 42 45  4d 44 4a 43 41 43 41 43   NEJEDEBE MDJCACAC
0050  41 43 41 43 41 43 41 41  41 00 20 45 44 45 49 45   ACACACAA A. EDEIE
0060  46 45 4e 45 4a 45 44 45  42 45 4d 43 41 43 41 43   FENEJEDE BEMCACAC
0070  41 43 41 43 41 43 41 43  41 42 4e 00 ff 53 4d 42   ACACACAC ABN..SMB
0080  25 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   %....... ........

File: "/tmp/etherXXXXDSiO4" 72 KB ...   Packets: 140 Displayed: 140 Marked: 0 Dropped: 0   Profile: Default
```

Figure1: Exporting files from Wireshark

**Plain- text, space separated format:**

```
No     Time      Source              Destination         Protocol
1 10:09:22     192.168.3.96     192.168.255.255     TCP
2 10:09:22     192.168.3.96     192.168.255.255     NBNS
3 10:09:22     192.168.3.117   192.168.255.255     NBNS
4 10:09:22     HewlettP           Broadcast               ARP
```

**CSV files**

```
No     Time      Source                  Destination       Protocol
“1”, ”10:09:22”, ”192.168.3.6”, ”192.168.255.255”, ”BROWSer”,
”2”, ”10:09:22”, ”192.168.3.6”, ”192.168.255.255”,”NBNS” ,”92”
”3”, ”10:09:22”,”192.168.3.7”, ”192.168.255.255”, ”NBNS”, ”92”
”4”, ”10:09:22”, ”HewlettP”, ”Broadcast”, ”ARP”,”60”
```

**tcpdstat[10]**

Written by Dave Dittrich, **tcpdstat** is a powerful tool that performs an in-depth protocol breakdown by bytes and packets. It further displays average and maximum transfer rates, IP flow information, and packet size distribution.

| port | usage | Ip | usage |
|---|---|---|---|
| 80 | 33.80% | 131.243.89.214 | 8.50% |
| 22 | 16.70% | 128.3.2.102 | 6.20% |
| 11001 | 12.40% | 204.116.120.26 | 4.8 |
| 2049 | 10.70% | 128.3.161.32 | 3.60% |
| 1023 | 10.60% | 131.243.89.4 | 3.50% |
| 993 | 8.20% | 128.3.164.194 | 2.70% |
| 1049 | 8.10% | 128.3.164.15 | 2.40% |
| 524 | 6.60% | 128.55.82.146 | 2.40% |
| 33305 | 4.50% | 131.243.88.227 | 2.3 |

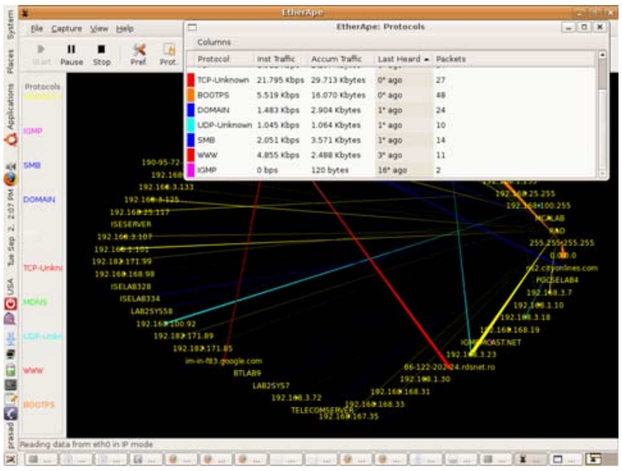
**tcptrace [12]** is another program which will sniff network and monitor data and report the events, as given above.An example output taken from **[11]** is as below

```
DumpFile:  trace.pcap
FileSize: 98876.89MB
Id: 200703011241
StartTime: (anonymized)
EndTime:   (anonymized)
TotalTime: 7216.13 seconds
TotalCapSize: 96826.91MB  CapLen: 1514 bytes
# of packets: 134347439 (96826.91MB)
AvgRate: 113.10Mbps  stddev:47.96M   PeakRate:
260.92Mbps

### IP flow (unique src/dst pair) Information ###
# of flows: 1612801  (avg. 83.30 pkts/flow)
Top 10 big flow size (bytes/total in %):
 33.6%  3.2%  2.2%  1.5%  1.4%  1.0%  1.0%  0.9%
0.8%  0.8%

### IP address Information ###
# of IPv4 addresses: 480065
Top 10 bandwidth usage (bytes/total in %):
 34.4% 34.4%  3.3%  3.3%  3.0%  2.7%  2.3%  1.8%
1.5%  1.5%

### Packet Size Distribution (including MAC
headers) ###
 [   32-   63]:   20839652
 [   64-  127]:   38798140
```

**Snort [13]** is currently a de-facto standard for NIDS which works mainly with the signatures and generates alerts according to the predefined rules or signatures, Again here using snort to log the alerts to the database is also not a good idea due to the limitations of the snort engine in updating the database, that is snort engine has a

limit over writing the records to the database per minute which makes it slow. Also Snort fails for a very high speed networks and when the buffers gets filled several packets will traverse without getting processed. Snort cannot generate statistics in such a way that the statistics that it generates is useful for identifying new attacks. Snort is useful only in signature
and specification based approaches and fails in case of bottleneck and anomaly detection approaches to detect attacks on host or a network. The other option Snort provides is to log the alerts or packets to a directory which again is one more step slower where the logs have to be scanned for traffic generated by different machines and from that generating statistics is a bit longer process. Snort also logs the alerts to the file based on the rules written to monitor the network. From the log file information could be extracted but again they are based on rules predefined which don't detect any new attack. Snort allows writing preprocessors where each packet could be processed and generating alerts according to the need of a host or a network. Writing preprocessors is a tedious job and needs special developers since Snort is not completely well documented.

**Etherape [14]** is another network monitoring tool which gives statistics about the traffic over network visually, But etherape doesn't allow to log the traffic to a file but only gives a pictorial representation of the traffic flow over a network as given in the figure below.



| Protocol | Inst Traffic | Accum Traffic | Last Heard | Packets |
|---|---|---|---|---|
| 🟥 TCP-Unknown | 21.795 Kbps | 29.713 Kbytes | 0" ago | 27 |
| 🟧 BOOTPS | 5.519 Kbps | 16.070 Kbytes | 0" ago | 48 |
| 🟦 DOMAIN | 1.483 Kbps | 2.904 Kbytes | 1" ago | 24 |
| 🟦 UDP-Unknown | 1.045 Kbps | 1.064 Kbytes | 1" ago | 10 |
| 🟦 SMB | 2.051 Kbps | 3.571 Kbytes | 1" ago | 14 |
| 🟥 WWW | 4.855 Kbps | 2.488 Kbytes | 3" ago | 11 |
| 🟪 IGMP | 0 bps | 120 bytes | 16" ago | 2 |

**Figure 2** : etherape monitoring snapshot

## 5. Required format of information for statistical traffic anomaly detection

From the format of the information given in the above Table1 it will be very easy to draw conclusions about the normalcy of traffic in a network. Also the pictorial representation of the network activities will help in knowing the deviations in less time. Since the statistical anomaly detection techniques need most of the time numbers of different parameters.

| protocol | T1 | T1+ 10 | T1 + 20 | T1 + 30 | T1+ 40 | T1 + 50 |
|---|---|---|---|---|---|---|
| loop | 57 | 45 | 7 | 12 | 23 | 56 |
| llc | 112 | 233 | 66 | 12 | 235 | 23 |
| ntp | 232 | 333 | 643 | 435 | 676 | 132 |
| arp | 112 | 321 | 345 | 454 | 8 | 214 |
| icmp | 123 | 54 | 34 | 656 | 99 | 31 |
| dns | 232 | 45 | 556 | 5 | 98 | 334 |
| tcp | 223 | 665 | 56 | 567 | 6544 | 433 |
| ssl | 2 | 57 | 88 | 668 | 23 | 46 |
| telnet | 23 | 68 | 89 | 78 | 34 | 44 |

T1:1998-06-05 17:20:32.068 - UTC format
**Table 1**: Format of Statistical report required

| Source IP | Protocol | No of Packets |
|---|---|---|
| 131.84.1.31 | TCP | 314 |
| 134.205.165.12 | HTTP | 18 |
| 134.205.11.22 | TCP | 342 |
| 135.13.2.16 | SMTP | 23 |
| 135.13.2.17 | TCP | 50 |
| 135.14.2.18 | TELNET | 33 |
| 136.33.12.5 | TCP | 35 |

**Table 2**: Brief statistical report of SourceIP

The information given in the above Table2 will be useful to know about a particular computer represented by the IP regarding the activities it is initiating. This type of information will help to analyze the behavior of a machine. By combining Table1 and Table2 it is possible to make useful conclusions that help in detecting the attacks.

### 5.1 Information Extraction Challenges

#### 5.1.1 Extraction using DDL, DML

While extracting the information using DDL, DML [20] on database to generate statistics about the traffic and also generating the statistics about the user behavior or any other program behavior is very time consuming. After

inserting the records into the database, retrieval is not an issue but finding the patterns and creating views of the patterns in the databases will take huge amount of time and memory.
For example if the database has **m** records and **n** columns and to generate statistics as given in the Figure **[3]** requires **m-1 x n-1** update query executions, which works fine with less number of records. But the network data is a very large database ranging from at least few GB to several hundred of GB per day in a network of 100 machines. This is a huge data.

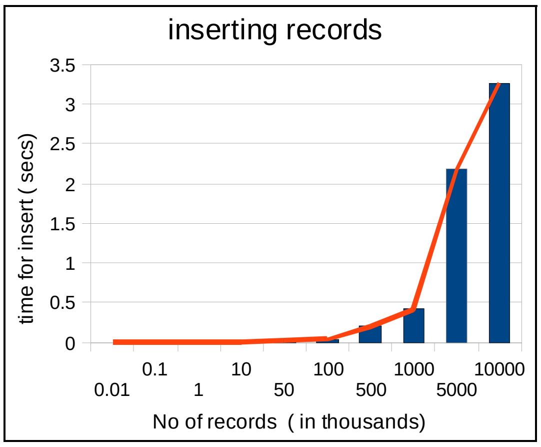


**Figure 3**: Time taken for inserting records into mysql database

**Figure 4** explains more about the time taken for updating **m** rows and only **one** column which reveals that it is inefficient to use databases to create views or create statistics from which any intrusion detection could be carried out. Our approach has an advantage over using just SQL (DDL and DDL or procedural) statements to generate statistics in terms of time and space

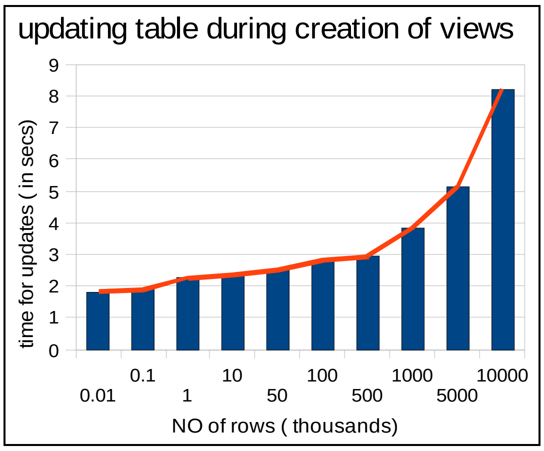


**Figure 4**: Time taken for updating tables recursively for many rows and one column

## 5.1.2 Extraction using gawk / awk / sed

While extracting the information using gawk[16] / awk [15] / sed[17] alone for extracting information from the network data which are exported to a plain text space separated files using wireshark will yield faster results only up to certain number of records after a certain number of records and also depending upon the main memory and the processor speed scripts written in gawk and sed will not give the results faster. **Table [3]** and **Figure [5]** gives an idea about the time taken to generate statistics from network data for traffic anomaly detection.
When the number of records crosses 25,00,000 records CPU and memory requirement will increase too much which is the major drawback of using scripts written in gawk /sed.

| No of rows | T=15 secs | T=10 Secs | T=5 Secs |
|---|---|---|---|
| 1 | 0.01 | 0.01 | 0.1 |
| 5 | 0.02 | 0.02 | 0.2 |
| 10 | 0.03 | 0.03 | 0.3 |
| 25 | 0.04 | 0.04 | 0.4 |
| 50 | 0.1 | 0.1 | 1 |
| 100 | 0.4 | 0.5 | 1.5 |
| 250 | 3.5 | 5 | 6 |
| 500 | 8.5 | 10 | 15 |
| 1000 | 15 | 23 | 47 |

**Table 3**: Time taken to generate statistics by gawk scripts with more than one time slots (columns)

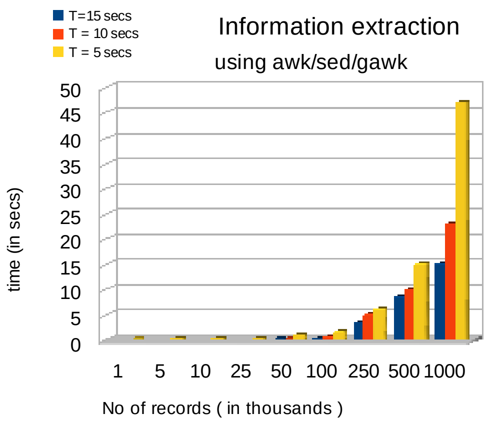


**Figure 5**: Time taken to generate statistics by gawk scripts with more than one time slots.

## 6. Our approach

our approach towards extracting information efficiently from the libpcap compatible

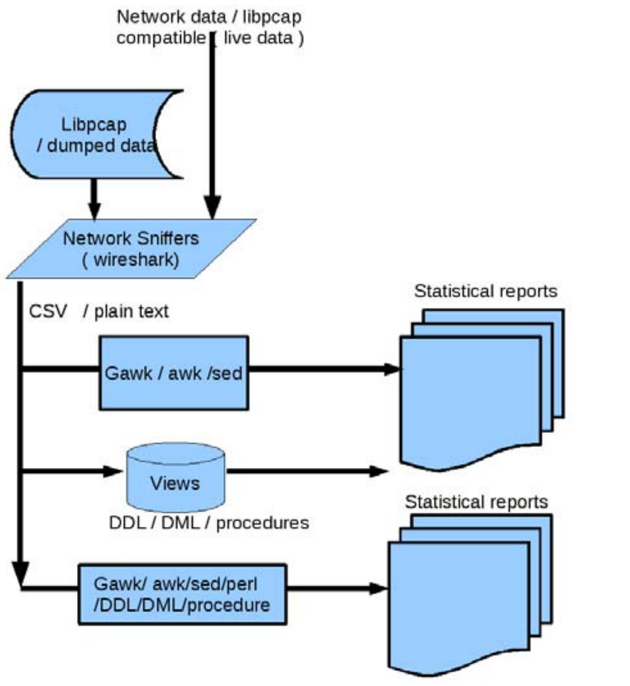


**Figure 6**: Our approach for extracting information

network data which were exported as space separated plain text or csv files using wireshark will be useful in detecting new attacks especially using traffic anomaly detection, specification based and bottleneck verification techniques.

When the network data is collected using a sniffer like tcpdump, the data will be in the

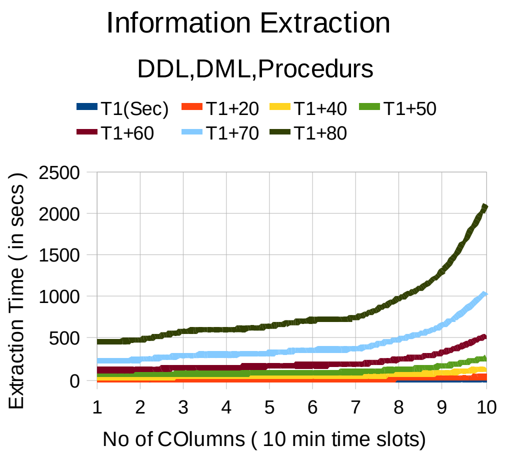
Figure 5: Time taken to generate statistics by gawk scripts with more than one time slots

binary format which is libpcap compatible. The DARPA98 dataset also contains tcpdump of seven weeks data as training data and two weeks of testing data. To extract information which will be in a format useful for traffic anomaly detection, three techniques are used. First technique uses gawk/ awk/ sed and the second technique makes use of database management systems and DDL,DML and procedural statements[20]. Finally third technique mixes both scripting and DDL/DML statements. Experiment results shows that the third technique performs better than the first two techniques.

## 7. Discussion and Results

Figure3 and Figure4 tells about the time taken in case of only one slot of 10 mins, which is only one column and many rows. When the number of columns increases, creating views, generating statistics as given in Table2 and Figure5 will take lot of time and after a certain limit the time taken is so huge that even high speed computers will fail to complete the tasks.

To generate similar statistics as given in Table1 and Table2 using gawk / awk /sed will take less time compared to DDL or DML statements. But major disadvantage of using gawk /awk /sed is that for more than 2500000 records and for time slots more than 250 it will be very very slow and many times processor will be completely busy and system hangs even on a system with



more than 4 GB RAM.

**Figure 7** : Time take to extract information using DDL/DML for more than one time slots

But when perl**[19]** / gawk / sed /awk are mixed with pager/ procedures / DDL /DML statements using database then the statistics will be generated much efficiently than just using DDL /DML / perl / gawk /awk /sed alone.

We see from the Figure 8 and comparing this with Figure3, Figure4, Figure5 and figure7 implies that our approach using scripts along with sql statements will give efficient results than just using scripting or sql statements alone.

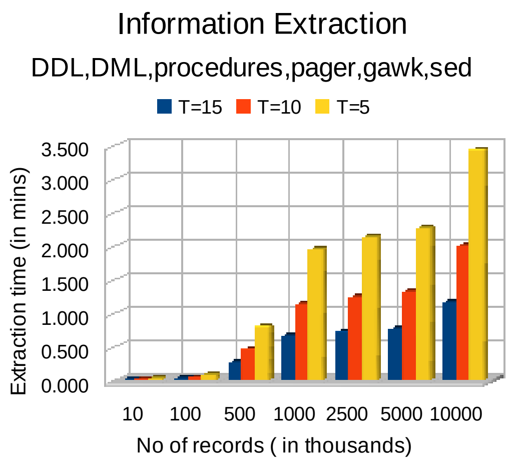


**Figure 8** : Time taken to extract information using DDL /DML /procedures / perl/gawk /sed.

## 8. Conclusions / Future Work

It is found that using DDL/DML/procedures along with perl/gawk and sed scripts are more efficient than using alone to extract information and generate statistical reports which are useful for statistical traffic anomaly detection. In future we will be using the reporsts generated by our approach and use it for offline and also online traffic anomaly detection.

### Acknowledgements

I thank the Principal and R V College of Engineering administration for all the encouragement and support extended to me in carrying out my research work.

### References:

[1] 'Computer Network Security' By Joseph Migga Kizza Published by Springer, 2005,IEEE ISBN 0387204733.
[2] Computer System Attack Classification N. Paulauskas, E. Garsva, Vilnius Gediminas Technical University, Department of Computer Engineering,
[3] An intrusion detection model by Dorothy E. Denning ,1986
[4] A Practical Guide to Managing Information Security By Steve Purser Published by Artech House, 2004 ISBN 1580537022
[5] http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/1998data.html
[6] Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation, Lippmann, R.P.; Fried, D.J.; Graf, I.; Haines, Kendall, K.R, DARPA Information Survivability Conference and Exposition, 2000.
[7] A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems by Kristopher Kendall
[8] http://tcpdump.org
[9] http;//wireshark.org
[10] http://staff.washington.edu/dittrich/talks/core02/tools/tools.html
[11] http://matthias.vallentin.cc/2007/01/examining-and-dissecting-tcpdump-libpcap-traces/
[12] http://www.snort.org
[13] http://etherape.sourceforge.net/
[14] "Effective awk programming" by Arnold Robbins, 3rd edition, O'reilly
[15] www.gnu.org/software/gawk/gawk.html
[16] http://sed.sourceforge.net/
[17] http://sourceforge.net/projects/libpcap/
[18] http://www.perl.org
[19] http://dev.mysql.com/doc/refman/6.0/en/index .html

Renuka Prasad B is currently working as Lecturer in MCA department of R.V.College of Engineering, Bangalore, Karnataka,India pursuing his Phd in Dr MGR Educational and Research Institute, Tamil Nadu,

Prof. Dr Annamma Abraham is currently working as Professor in MCA department of R.V.College of Engineering, Bangalore, Karnataka, India.

Chandan.C is a student of R.V.College of Engineering, Bangalore, Karnataka, India pursuing his Master of Coputer Application and currently is in the final year of his course.

Prabhanjan.A is a student of R.V.College of Engineering, Bangalore, Karnataka, India pursuing his Bachelors Degree in Computer Science and Engineering and currently is in the fourth year of his course.

Ajay Bilotia is a student of R.V.College of Engineering, Bangalore, Karnataka, India pursuing his Master of Coputer Application and currently is in the second year of his course.