Usage of Netflow in Security and Monitoring of Computer Networks

Shivam Choudhary
MIT Manipal

Bhargav Srinivasan
MIT Manipal

ABSTRACT
Management of a network is a challenging task without accurate traffic statistics. Through this paper the security benefits of implementing a Netflow [1] based analysis system and then a novel open source application useful in Netflow analysis and management of flow records is proposed. Netflow data provides important information about network conversations and behavior. Netflow statistics are generated by Cisco and Juniper routers and switches, as well as server software Netflow probes. Netflow data provides enough information to serve the needs of several different applications such as billing, network planning and most importantly traffic engineering, which we specifically analyze to assess the state of the network. The flow records are UDP packets lacking payload data which still provides enough data to the network administrator to be a valuable analysis tool. Netflow profiling is a good moderation which strikes a balance between detail and summary and provides a real-time analysis of traffic flows, connection information and abnormal network behavior. Netflow data on the router is sampled at a variable rate which actually satisfies the conditions which have been set to monitor the incoming traffic, and then compared periodically to test if an incoming sequence is a DOS or a possible threat to the network by examining the packet sequences. Through this paper the performance of a network is gauged by using a parameter called unexpectedness [2], which is a method to gauge the amount of traffic flowing through a network.

General Terms
Netflow, Virtual Router, GNS3, Cisco Packet Tracer, Backtrack

Keywords
Netflow, Cisco Packet Tracer, GNS3, Virtual Box, Virtual Router, Unexpectedness, Decoding Netflow Packets.

1. INTRODUCTION
Netflow [1] is a traffic profile monitoring technology developed by Darren Kerr and Barry Bruins at Cisco Systems, back in 1996. Netflow data provides important information about network conversations and behaviour. Each unique flow is recorded by the network devices or probes, and the flows are then reported to a data collection server.

The notion of Net*flow* [3] *profiling* was introduced within the networking research community, and subsequently extended by other researchers, to monitor internet traffic. Netflow profiling had been predicted to be relevant to applications such as route caching and usage-based accounting i.e. a method of billing customers. Today, based in part upon market demands for performance and accounting, Netflow profiling is built into networking devices. Although proper standards have not yet been established, Netflow methodology is robust enough to persist through this period of vendor-specific implementations, and its benefits and popularity in the industry warrant its early adoption.

Table 1: Certain important fields encapsulated in the flow records of a Netflow v5 packet

Name	Description
srcaddr	Source address
dstaddr	Destination address
input	Input interface
output	Output interface
dPkts	Number of packets
dOctets	Number of octets
First	Start of NetFlow
Last	End of NetFlow
srcport	Source port
dstport	Destination port
tcp_flags	TCP flags
tos	IP type-of-service

Network administrators who collect measurement data often find that they either have collected too little data or too much of it. In a sense, Netflow profiling is a good moderation which strikes a balance between detail and summary. The proposed method is network independent so it can adjust to the requirements of any kind of network. This type of data proves invaluable against DoS (denial of service) attack and overloading of the network. In this paper a system which can be effectively used to capture Netflow Data and can be used to extract certain parameters from it is described so that a DoS attack can be detected and prevented.

1.1 Netflow v5 Flow Record Format
As described by Table 1, the Netflow v5 packet received can be decoded using a Perl script which we designed, with reference to certain previously designed modules which effectively captures and separates the various fields. This script also logs the data which can

then be used for further processing. An algorithm which can be used to design a system for processing the sampled flow data has been proposed.

2. Outline of the System

The monitoring system comprises of a router which is configured to export Netflow information to a flow database which is organized by a Perl script. The script then stores the flow information in a separate file every day and a separate folder for each month. The Netflow data is then compared periodically to test if an incoming sequence is a DOS or a possible threat to the network by examining the packet sequences that arrive through the router. This is clearly depicted in Figure 1.

The monitoring station is a small machine that reads the exported flows and compares them to an existing malicious packet sequence database. When a sequence of malicious packets is matched and detected, then the system generates an alert status. This could then be further analysed by network administrators to determine whether the threat needs attention.

The router is configured to store flow information, by the user, and it transmits this information to a database which is listening for it. In the proposed simulation case, the router sends the Netflow v5 information in the form of UDP packets, through a specific port (2055 in test case), to a client which is listening on the same port. Once the packets are received, Perl modules [4] are used which help decode the packet and extract flow information. The script reads the incoming UDP packets and extracts the flow headers and the required flow records and stores the information in a log file. A database is created using a Perl module which reiterates his function periodically, forming new data logs on a daily, monthly and yearly basis while organizing them accordingly.

A small Perl snippet (figure 2) was created as required and it can be easily modified to provide efficient methods of organization for the logged and pre-processed data. The records are indexed effectively and an optimal searching algorithm can be formulated to effectively retrieve the required flow data. The available Netflow data on the router is sampled at a variable rate of Z samples per second which corresponds to the cluster of packets that actually satisfy the conditions which have been set to monitor the incoming Netflow packets, and only these ones will be allowed to enter the processing system, which will then perform a detailed analysis of the incoming traffic and check for any anomalies.

As depicted through the flowcharts (figure 3), the monitoring system borrows from the database the values of previously identified attack patterns and sequences, which are simultaneously compared with incoming traffic. This system monitors traffic on ports which are most susceptible to attacks. Effectively the monitoring system acts as an interface between two databases i.e. the collection database and the Threat database.

2.1 Monitoring Station

The role of the monitoring station is to judge whether the incoming traffic requires the attention and detailed threat analysis using the system. The station is designed to segregate packets which flow into the processing module and those which are just sent for storage. Condition of Denial of Service (DOS) attacks or network overloading can be established by evaluating if either one or more of the following signatures are observed.

2.1.1 Malicious Data Packets

The following are the examples of how the "malicious" data packets can look like and how one can configure the router to protect against the specific type of attack. Finally, at the end, methods consolidate all the protection mechanisms into a single firewall configuration and use it to protect the system against DOS attack is proposed.

2.1.1.1 ARP poisoning:

There may be improper configuration in the network causing broadcast storms and lot of ARP packets might be received by the router. Also, sometimes it is possible that a customer facing port receives lot of ARP request and reply packets as part of a DOS attack. Thus a simple rate limit imposed on such packets can be an effective prevention strategy.

2.1.1.2 Packets with strange (Martian) addresses/ strange IP options field:

They commonly are sent by improperly configured systems on the network and have destination addresses that are obviously invalid. Malformed packets such as these are most likely crafted and are often part of DOS attacks. Hence a filter is set in order to check for martian IPs. IPs with options field enabled are not detected by the router, hence an IP packet with too many options can easily be used to fabricate a DOS attack. These types of packets are strictly monitored and filtered to protect system resources from being depleted or misused.

2.1.1.3 Malicious control plane packets:

A router may be subject to DOS attacks by sending large number of such packets which go up to the router. Such control packets generated by a DOS attacker will contain either invalid information that will be discarded, or may even contain malicious information that can cause router's forwarding information to get corrupted.

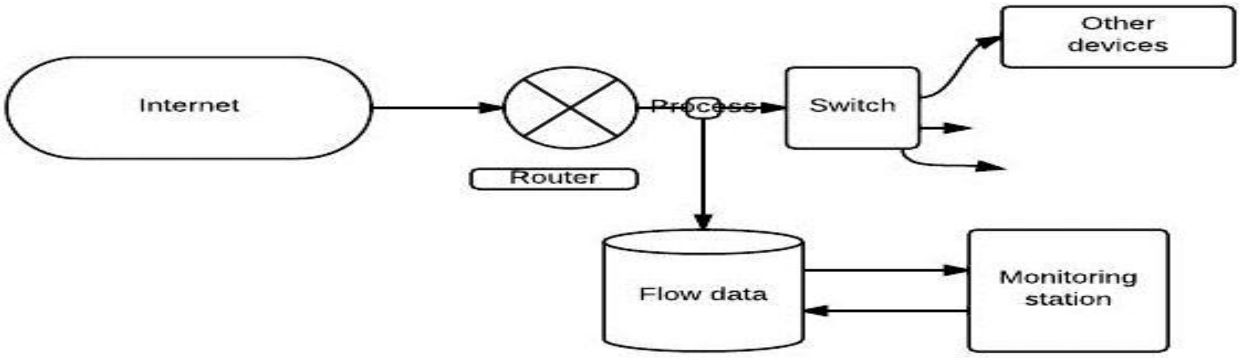

Figure 1: Diagram showing the logical flow of packets

Perl program snippet to store Netflow in an organized manner.

```
$now_time = strftime "%Y%m%d", localtime;
$now_month = int($now_time/100);

if($now_time!=$first_time || !$run) {               # check if date (year,month,day) changed
if($now_month!=$first_month || !$run) {     # if date changed check if month changed
$path = 'd:\perl\logm' + $no + '\log1.txt';    # create a new monthly table
$first_month=$now_month;
$no = $no +1;
}
$path = 'd:\perl\logm' + $no + '\log' + $no2 +'.txt'; # create table entries for today
$first_time=$now_time;
$run = 1;                    # from now on we only check for the tables when the date changed
$no2 = $no2 +1;
}
```

Figure 2: Perl Program to store the Netflow Data

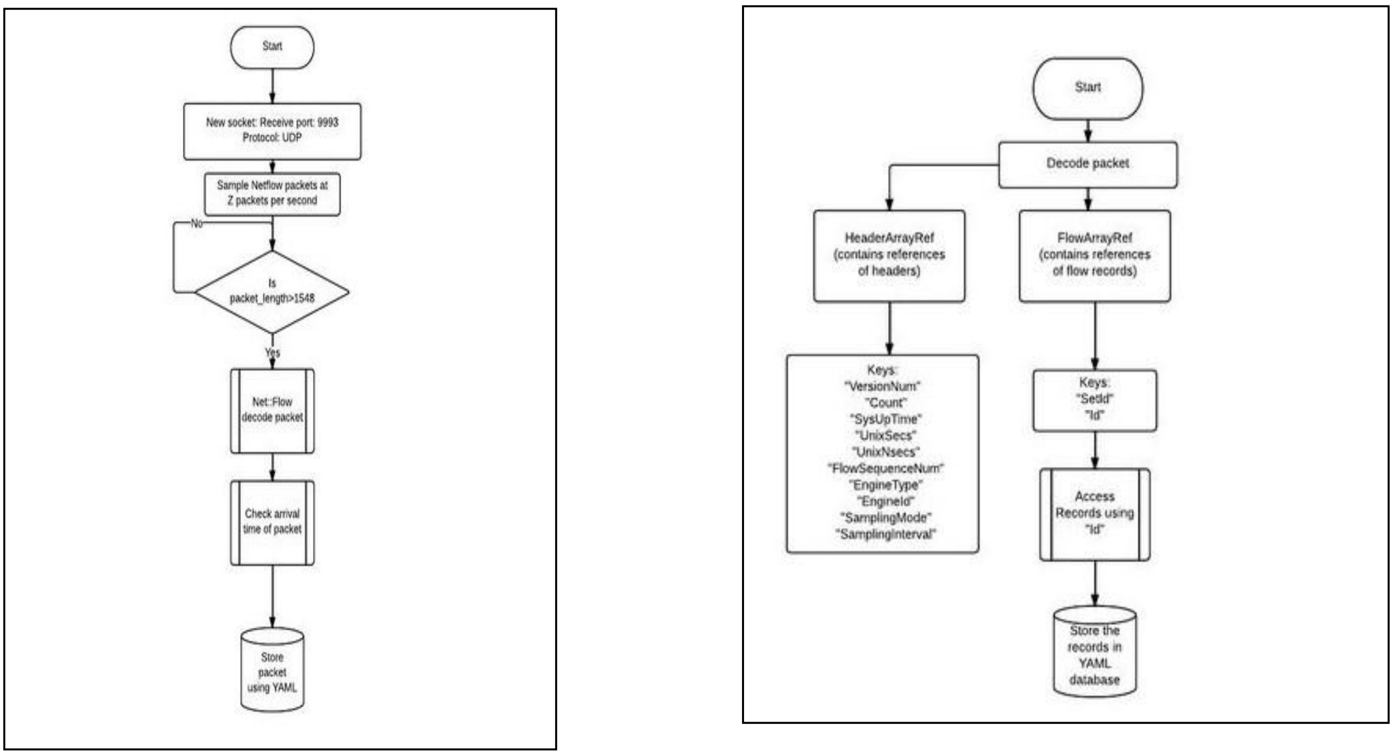

Figure 3: Flow Diagram showing storage and collection of Netflow Data Records

Control plane packets are accepted only from well-known peers and thus such a filter prevents injection of routing protocol PDUs into the network and hence tampering with the routing configuration.

2.1.1.4 Half open TCP ports (TCP SYN/RST, RST/FIN attacks):

A very common DOS attack experienced by routers is the TCP SYN/RST attack. Here a lot of malicious TCP SYN packets or RST packets are received by a router. Now, on receiving a TCP SYN packet, the system will attempt to complete a TCP 3- way handshake, and will queue up the TCP SYN packets in its protocol control blocks as pending connections. This leads to the port being half open and later become an unused port. This is one of the very popular methods for performing a DOS attack, and is preferred by most attackers or script kiddies. Similarly, when it receives a packet with RST/FIN, it will queue them up for processing and it leads to half open ports, resulting in a scenario similar to the one previously described. Thus a rate limiting system can be implemented to provide a check and thus aid in the prevention of such malicious activity.

2.1.1.5 Flood of UDP/ICMP and other OAM packets:

An attacker usually tries to deplete all network resources by flooding the system with ICMP pings or trace routes or other types of OAM packets. Therefore a rate limit has to be brought into effect, in order to prevent overloading. Other forms of injected packets overloading such as ftp, SNMP, ntp, DNS can also be capped.

Figure 4 shows the proposed algorithm to tackle all the idiosyncrasies of the packets.

3. Formulation of Unexpectedness

Through the simulations, the performance of a network is gauged by using a parameter called unexpectedness [2], which is a method to gauge the amount of traffic flowing through a network. The value can then be used to infer traffic deviation from pre-set 'normal' conditions, and hence the flow of traffic through the network can be monitored and it provides the administrator with a profound understanding of the network's current state.

The value of unexpectedness is a standardized value, which is network independent and can function as a tool to monitor the severity of overloading of the network. The value gives administrators a fair idea about the traffic flowing through the network, and thus enables appropriate judgement.

3.1 Proposed Algorithm

In the algorithm an extremely simple method based on the available flow data to calculate the unexpectedness and it involves defining the following parameters,

U = Rate of sampling and storage of processed flow records at a variable rate

L_a = Rate of incoming flow records (Flows/sec) at the router in an "attack" scenario

L_n =Rate of incoming flow records (Flows/sec) at the router in a "normal" scenario

L_{neff} = the effective rate of incoming flow records in a "normal" scenario

L_{aeff} = the effective rate of incoming flow records in an "attack" scenario

$L_{neff} = U*(L_n/ (L_n+L_a))$ under normal conditions

$L_{aeff} = U*(L_a/ (L_n+L_a))$ under attack conditions

Let L_n = 10^4; L_a = 10^5; U = 2·10^4 ;(we leave out the units for better readability)

We obtain L_{neff} = 1.82 · 10^3 and L_{aeff} = 1.82·10^4

Now, if the attack rate increases by 10% then L'_a = 1.1·10^5,

The effective rates become L'_{neff}= 1.67·10^3 and L'_{aeff} = 1.83 · 10^4,

i.e., the effective rate of attack flow records increases by 0.83%,

While the effective rate of normal flow records decreases by 8.3%

So, Unexpectedness = L'aeff x 100% = 83%.

Hence this raises an alarm and prompts the network administrator to further investigate the issue.

3.2 Result of the above formulation

In order to obtain the above calculations, we have considered two scenarios a 'normal' scenario and an 'attack' scenario. The 'normal' scenario entails computers to operate in a fully functional optimally loaded state, which is considered as regular and untampered behaviour which is carefully judged and pre-set by the administrators according to their requirements. The 'attack' scenario is characterized by all the systems overloading the network such that it emulates a denial of service attack or simply depletion of network resources. Hence the unexpectedness scale can be obtained to monitor the network effectively and this approach makes this system scalable to a very high extent.

4. Simulation of the Network

Simulation of the network was achieved using a complete virtualization of the system, provided by the popular network simulation tool GNS3. Using the IOS

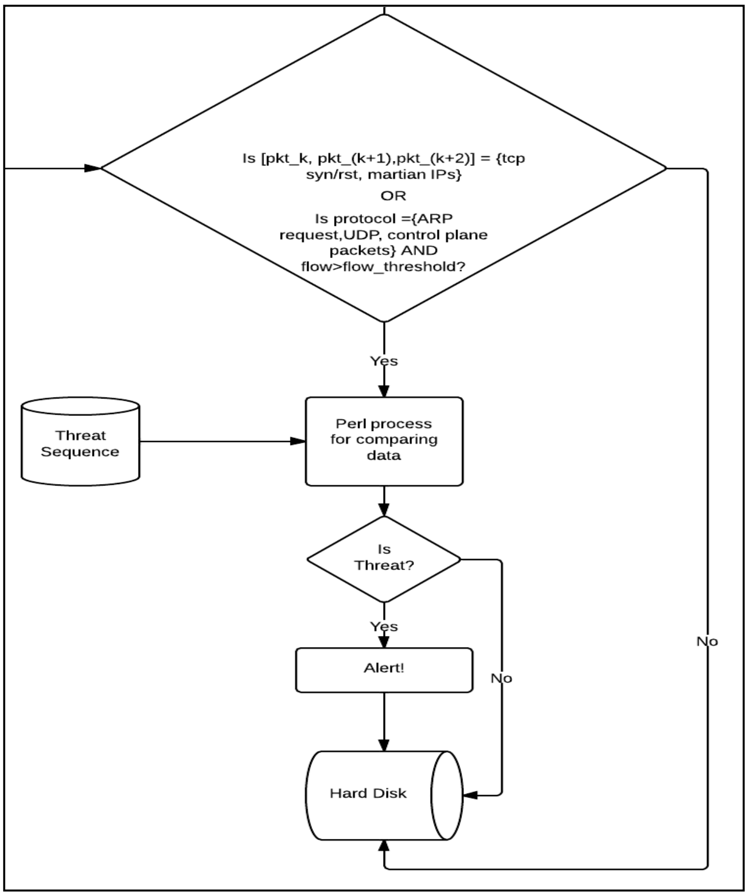

Figure 4: Flowchart illustrating a proposed algorithm for the monitoring station.

image of a Cisco 3620 series router the virtual test network was created. We start by having two networks in which one is the administrator and his router is having Netflow enabled on it and the other is the attacker which is trying to attack the administrator's network, and we have configured two virtual machines to act as two independent hosts. The Backtrack R3 is a packet generation system and we have another Backtrack R3 which is the administrator's network. We are using Scapy [5] for packet crafting and transmission because it gives us more control over how the packet moves and more control over the various flags.

4.1 Configuration of the Network

The network is configured in the following way,

1) IP Address 10.1.5.100 (255.0.0.0) to the Backtrack Administrator system.

2) IP Address 192.168.44.1(255.255.255.0) to the Backtrack R3 packet injector System.

A router can be used to connect these two networks by configuring it properly. The topology of the network is as shown below in figure 5. The following topology was simulated in GNS3. The two clouds C1 and C2 represent two networks from Virtual Box. So effectively two machines running in Virtual Box are connected through a Virtual Router running in Windows.

4.2 Storing the Netflow Data

Netflow is enabled on the virtual router and then the Netflow data is dumped into the UDP port number 2055 of the admin machine (IP address 10.1.5.100). On this port a Perl script is running and this script is used to decode the packets and arrange them properly in months, days, years so that the data packets are properly arranged for future reference. The idea of the system is to analyse the Netflow data and then compare it with the records and look for certain parameters in the data so that an attack packet can be isolated and the network is saved from further harm. Statistically it is seen that an attacker or hacker never attacks all of a sudden. They keep on gathering information about a network for some time using custom scanning techniques which can be very discrete and cannot be prevented otherwise. For example if the network imposes a limit on the number of ports to be scanned a clever work around can be using an ever changing IP Address spoof and then keep on scanning the ports or a second and more crude way could be having a script scanning few ports (well within the limits of the network) without ever being detected.

4.3 Decoding of Netflow packets

This is the central problem of which we are trying to find a solution. So using Netflow all data packets are logged on and details are extracted. So for example if an attacker tries to attack the system still using different IP addresses but the behaviour of the packets matches a malicious pattern in the database then he can be suitably tracked down and preventive measures can be taken. If Netflow is enabled and the dump of the file is checked though we will get the file but we won't be able to decode anything from the dumped data. So proper decoding of the data is required to get the information from the Netflow packets. So in the system a decoding script is running on the port 2055 (in simulation case ip address 10.1.5.100) that decodes the data and dumps it into a text file. Figure 6 shows the Netflow data being dumped on the port without any decoding. So as one can infer from the figure it is gibberish and doesn't make any sense. But once the packets are properly decoded very useful parameters can be extracted from these packets.

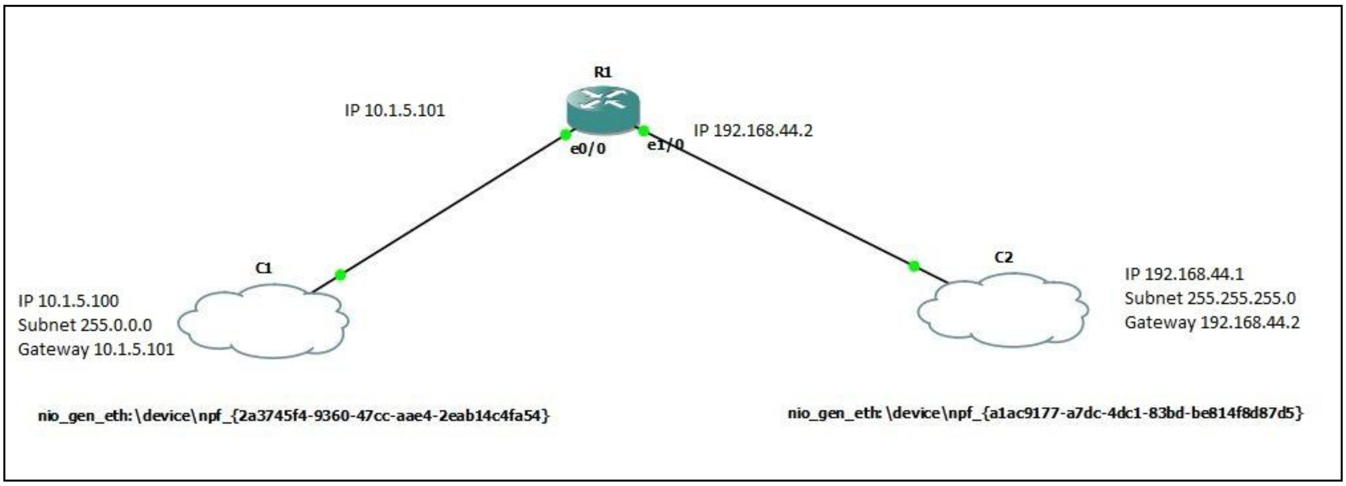

Figure 5: The Topology of the Network

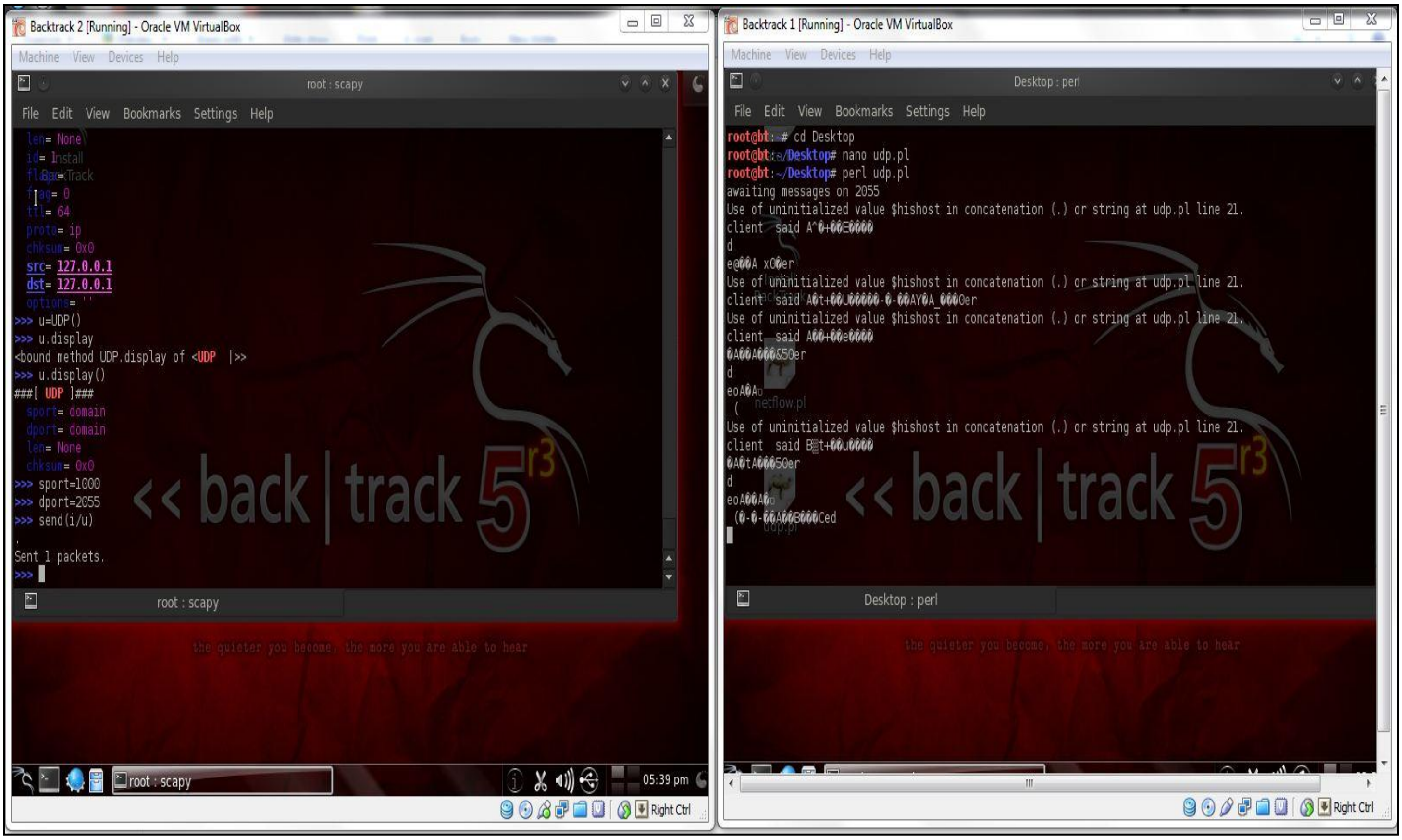

Figure 6: Captured packet format by Netflow

So from Figure 6 it can be inferred clearly that decoding of these packets is required to get the information. So a decoding script is running on port number 2055.So any data that gets dumped on port number 2055 is decoded and can be seen through the script.

To enable Netflow Version 5 on the router standard IOS commands is used. Also the port number and destination address to dump the UDP Netflow data is to be specified while configuring the router. Figure 7 clearly shows the router configuration. Just to check if the network is working properly it is pinged using ICMP packets. This result is summarized in Figure 8. From this it is obvious that the network is connected and the router is aware of it. These ICMP

```
interface Ethernet0/0
 ip address 10.1.5.101 255.0.0.0
 ip flow ingress
 ip route-cache flow
 half-duplex
!
interface Ethernet1/0
 ip address 192.168.44.2 255.0.0.0
 half-duplex
!
no ip http server
ip flow-export source Ethernet0/0
ip flow-export version 5
ip flow-export destination 10.1.5.100 2055
ip classless
 --More--
```

Figure 7: Router Netflow Configuration

packets just ensure that the network is present and does not tell us any information about the network. It is simply an indication of presence of a host computer on a particular IP address. Many network administrators (such as that of Apple) disable this because they unnecessarily engage the server with hello packets.
Also if the identity of the server is to be restricted to public then ICMP packets can be disabled on the network.
But in the test case network no Access Control List (ACL) is applied and no ports are blocked. So it can be seen clearly from figure 8 that one can easily ping the networks in the topology. Now Scapy [5] was used to generate packets and then send the packets to the Admin's network. Now through the router each and every packet will be captured and it will be dumped onto the Admin's side. So one can now have scripts running to compare the data packets in the network.

After the data has been dumped it will be decoded and then compared using another script to extract the details for that packet. And then these details will be used to determine whether the attack is happening or not. Figure 9 below shows those typical Netflow packets.

5. Conclusion

In this paper we set out to gain insights into the capabilities of Netflow. We can gain the same or additional information from Netflow compared to other measurement techniques be it more detailed, such as at a packet level, or more brief. We propose an algorithm and present an implementation that is able to assess and prevent any threat risks for the network using Netflow data.
The implementation of the system can be effectively summarized as an independent real time sampling, aggregating and monitoring system. The versatility of the system lies in the fact that it can be used to properly gauge the possibility of a DoS attack or depletion of resources using a scale based on formulating a standard value of unexpectedness.

Furthermore the system is in very early stage of its development and requires further research and improvement. Filtering of packets can be implemented more effectively using advanced techniques. The calculation of unexpectedness can be further improved and also the fact that Netflow loses certain amount of accuracy compared to SNMP (Simple Network Management Protocol) and TCP connection summaries, but it is a fair tradeoff in processing speeds, indexing capabilities of the flow records.
The methodology shows that the acquired Netflow summaries provide invaluable insight of the actual state of the network. Netflow is the weapon of choice for network administrators if right aggregation technique is employed.

```
R1#ping 10.1.5.101

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.5.101, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
R1#ping 10.1.5.100

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.5.100, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 8/12/20 ms
R1#ping 192.168.44.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.44.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 8/10/12 ms
R1#ping 192.168.44.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.44.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
R1#
```

Figure 8: Ping Results of the Network

```
R1#show ip cache flow
IP packet size distribution (4364 total packets):
   1-32   64   96  128  160  192  224  256  288  320  352  384  416  448  480
   .000 .574 .368 .027 .000 .016 .003 .009 .000 .000 .000 .000 .000 .000 .000

    512  544  576 1024 1536 2048 2560 3072 3584 4096 4608
   .000 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000

IP Flow Switching Cache, 278544 bytes
  1 active, 4095 inactive, 629 added
  12423 ager polls, 0 flow alloc failures
  Active flows timeout in 30 minutes
  Inactive flows timeout in 15 seconds
IP Sub Flow Cache, 17032 bytes
  0 active, 1024 inactive, 4 added, 4 added to flow
  0 alloc failures, 0 force free
  1 chunk, 1 chunk added
  last clearing of statistics never
Protocol         Total    Flows   Packets Bytes  Packets Active(Sec) Idle(Sec)
--------         Flows     /Sec     /Flow  /Pkt     /Sec     /Flow     /Flow
TCP-Telnet          22      0.0        42    41      0.0      12.5       6.0
UDP-other          432      0.0         7    61      0.1       3.9      15.4
ICMP               174      0.0         1   138      0.0       3.5      15.9
Total:             628      0.0         6    61      0.2       4.1      15.2
```

Figure 9: Netflow Data showing various protocols and their flow rates

6. REFERENCES

[1] Cisco, Cisco IOS Net Flow Technology Data Sheet. <http://www.cisco.com/go/netflow>

[2] Vojtech Krmicek. GEANT3 JRA2 T4 Internal Deliverable. "Inspecting DNS Flow Trace for Purposes of Botnet Detection port scanning." 2011. Paper

[3] Cisco Systems Inc., "NetFlow Services and Applications -White paper," <http://www.cisco.com/warp/public/cc/pd/iosw/ioft/neflct/tech/napps_wp.h%tm>.

[4] Decoding Kobayashi, Atsushi. "CPAN RT." *Net::Flow*. Cpan.org, 2008. Web. 15 Dec. 2012. Brown, L. D., Hua, H., and Gao, C. 2003.

[5] "Scapy", an open source software for generating custom packets<http://oss.netboxblue.com/pug/scapy.html>.com /pug/scapy.html>.

[6] William Stallings, SNMP, SNMPv2, SNMPv3 and RMON 1 and 2, 1999.