Evaluation of the Capabilities of WireShark as a tool for Intrusion Detection

Usha Banerjee
Department of Computer Science & Engineering
College of Engineering Roorkee

Ashutosh Vashishtha
Department of Information Technology
College of Engineering Roorkee
Roorkee, India

Mukul Saxena
Department of Information Technology
College of Engineering Roorkee
Roorkee, India

ABSTRACT

This paper illustrates the functionality of Wireshark as a sniffing tool in networks. This has been proven by an experimental setup which depicts the efficiency of detection of a malicious packet in any network. Testing has been achieved through experimentation on a real time network analyzed by Wireshark. Inferences have been made which clearly depict Wireshark's capabilities highlighting it as a strong candidate for future development into a robust intrusion detection system. This paper highlights the working of Wireshark as a network protocol analyzer and also accentuates its flexibility as an open source utility to allow developers to add possible functionalities of intrusion detection devices in it.

Keywords
Data, Intrusion Detection, Sniffing, WireShark

1. INTRODUCTION

Recent years have witnessed a great surge in the usage of mobile devices. Thus, research in the areas of mobile and ubiquitous computing is of prime importance. Security is one of the primary concerns of users of such devices [1]. In any typical network, wired or otherwise, unlikely and unwanted entry of malicious users and/or malicious data packets are a major concern as far as the security of the network is concerned. Data packets are the basic entities of all communication systems. Security of a network thus implies security of the data packets. A data packet is the most basic block of communication involving a streamlined flow of its infinite other replicas in order to transmit information from one device to another. A data packet is contained in data segment that holds other information like the protocol being used, the destination hardware address etc. In a nutshell, the identity of any packet coming from any unreliable source can be detected by studying its contents. This study of detecting and only viewing the contents of a data segment and its packet is termed as packet sniffing and when a log of this information is prepared, the technique is called packet logging. A packet analyzer is a computer software or hardware that can intercept and log traffic passing through a digital network or part of a network. As data streams flow across the network, the sniffer captures each packet and eventually decodes and analyzes its content according to the appropriate specification.

This paper analyzes the process of packet sniffing and packet logging. Wireshark is a commonly available open source network protocol analyzer. In this paper we use Wireshark to study the functionality of a packet analyzer. Using Wireshark it becomes very convenient to detect any suspicious packet entry from any unreliable source. Any packet sniffer/logger with the added functionality of detecting malicious entries in a network is termed as an intrusion detection system (IDS) [2,3,4]. Furthermore, an IDS usually stores a database of known attack signatures and can compare patterns of activity, traffic or behavior it sees in the logs it is monitoring against those signatures to recognize when a close match between a signature and current or recent behavior occurs. At that point, the IDS can issue alarms or alerts. A signature is a pattern that matches a known malware. In this paper a testing problem has been designed and on the basis of the results of the experiment, appropriate conclusions have been draw indicating Wireshark's capabilities as a possible IDS.

2. A. Brief History of Packet Sniffers/Loggers

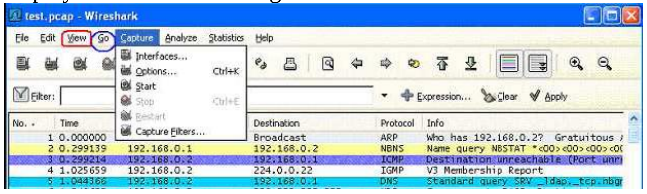
The goal of packet sniffing is to monitor network assets to detect anomalous behavior and misuse. This concept has been around for nearly twenty years but only recently has it seen a dramatic rise in popularity and incorporation into the overall information security infrastructure. Beginning in 1980, with James Anderson's paper [12], Computer Security Threat Monitoring and Surveillance, the notion of intrusion detection was born. James Anderson's seminal paper, written for a government organization, introduced the notion that audit trails contained vital information that could be valuable in tracking misuse and understanding user behavior. His work was the start of host-based intrusion detection and IDS in general. In 1988, the Haystack project [13] at Lawrence Livermore Labs released another version of intrusion detection for the US Air Force. This project produced an IDS that analyzed audit data by comparing it with defined patterns. In a telephone interview with the author, Crosby Marks, a former Haystack Project team member and Haystack Labs employee said that, "searching through this large amount of data for one specific misuse was equivalent to looking for a needle in a haystack." In 1990, UC Davis's Todd Heberlein introduced the idea of network intrusion detection. Heberlein [14] was the primary author and developer of Network Security Monitor (NSM), the first network intrusion detection system. Commercial development of intrusion detection technologies began in the early 1990s. Haystack Labs was the first commercial vendor of IDS tools, with its Stalker line of host-based products. Nonetheless, commercial intrusion detection systems developed slowly during these years and only truly blossomed towards the latter half of the decade. The intrusion detection market began to gain in popularity and truly generate revenues around 1997. Gerald Combs started writing a program called Ethereal so that he could have a tool to capture and analyze packets; he released the first version around 1998. The name was subsequently changed to Wireshark in May, 2006 owing to copyright issues.

3. WIRESHARK

Wireshark [2] is the world's most popular network protocol analyzer. It has a rich and powerful feature set and runs on most computing platforms including Windows, OS X, Linux, and UNIX. Network professionals, security experts, developers, and educators around the world use it regularly. It is freely available as open source, and is released under the GNU General Public License version 2. It has been developed and maintained by a global team of protocol experts, and it is an example of a disruptive technology. Wireshark formerly used to be known as Ethereal. Wireshark is a free packet sniffer computer application. It is used for network troubleshooting, analysis, software and communications protocol development, and education. In June 2006 the project was renamed from Ethereal due to trademark issues. Wireshark has tools for capturing, viewing, and analysis of data packets. Wireshark has sophisticated wireless protocol analysis support to help administrators troubleshoot wireless networks. With the appropriate driver support, Wireshark can capture traffic "from the air" and decode it into a format that helps administrators track down issues that are causing poor performance, intermittent connectivity, and other common problems.

4. SNIFFING TOOLS

Traditional network sniffing on an Ethernet network is fairly easy to set up. In a shared environment, an analysis workstation running Wireshark starts a new packet capture, which configures the card in promiscuous mode and waits until the desired amount of traffic has been captured. A node can be connected to a network through multitude of mechanisms, wired and wireless, covering many topologies and making use of wide variety of protocols. Wireshark provides users the capability of capturing the packets traveling over the entire network on a particular interface at a particular time. One of the primary tools is the capture tool. The interface option as shown in figure 1 below lists all available interfaces on the node and can enable capturing for any of these nodes. Options tab provides more sophisticated approach for each interface one at a time. The go menu items provide the capabilities of going through packets in the capture list. The View menu provides tools for listing packets, time display formats and coloring rules.

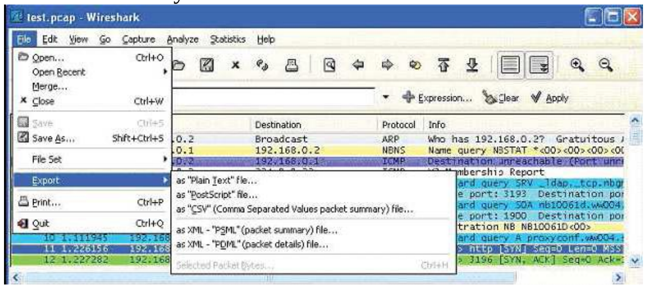


```
No. .  Time                          Destination     Protocol  Info
    1 0.000000                       Broadcast       ARP       Who has 192.168.0.2?  Gratuitous
    2 0.299139   192.168.0.1         192.168.0.2     NBNS      Name query NBSTAT *<00><00><00><0
    3 0.299214   192.168.0.2         192.168.0.1     ICMP      Destination unreachable (Port unr
    4 1.025659   192.168.0.2         224.0.0.22      IGMP      V3 Membership Report
    5 1.044366   192.168.0.2         192.168.0.1     DNS       Standard query SRV _ldap._tcp.nbgr
```

Figure 1. The Capture Tool

5. LOGGING TOOLS

Wireshark provides amazing flexibility over other IDS/IPS devices in the field of log maintenance. Log files can be captured at an hourly or weekly rate based on the requirement of the network and the capability of handling devices. Thus, files can be easily captured over a fast processing node and transferred to a slower database. Another interesting aspect is the feature of exporting the capture file into various other and more understandable formats- the plain text, post script, the CSV etc. based on the analyzer tool used.


test.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Help
Open... Ctrl+O
Open Recent
Merge...
Close Ctrl+W
Save Ctrl+S
Save As... Shift+Ctrl+S
File Set
Export
Print... Ctrl+P
Quit Ctrl+Q
as "Plain Text" file...
as "PostScript" file...
as "CSV" (Comma Separated Values packet summary) file...
as XML - "PSML" (packet summary) file...
as XML - "PDML" (packet details) file...
Selected Packet Bytes... Ctrl+H
Expression... Clear Apply

```
          Destination     Protocol  Info
.0.2      Broadcast       ARP       Who has 192.168.0.2?  Gratuitous
.0.1      192.168.0.2     NBNS      Name query NBSTAT *<00><00><00><0
.0.2      192.168.0.1     ICMP      Destination unreachable (Port unr
                                    mbership Report
                                    ard query SRV _ldap._tcp.nbgr
                                    e port: 3193  Destination por
                                    ard query SOA nb10061d.ww004.
                                    e port: 1900  Destination por
                                    tration NB NB10061D<00>
  10 1.111945   192.168             ard query A proxyconf.ww004.
  11 1.226156   192.168             > http [SYN] Seq=0 Len=0 MSS
  12 1.227282   192.168             > 3196 [SYN, ACK] Seq=0 Ack=
```

Figure 2. The Analyzer tool

5.1 Prefiltering and Analysis

Wireshark has two filtering languages: one used when capturing packets, and one used when displaying packets. Display filters allow to concentrate on the packets that the administrator is interested in, while hiding the currently uninteresting ones. Packets can be selected on the basis of protocol, the presence of a field, the values of fields, comparison between fields etc. The queries which can be entered inside the field or the expression tab (figure 3) can be selected to provide with much advanced definitions and listing all the protocols from wide range of protocols in Wireshark. Although only simpler commands can be used manually, few of these queries have been used in our experiment. The important feature of these commands is the use of operators, logical-and logical-or and negation operators are but a few to

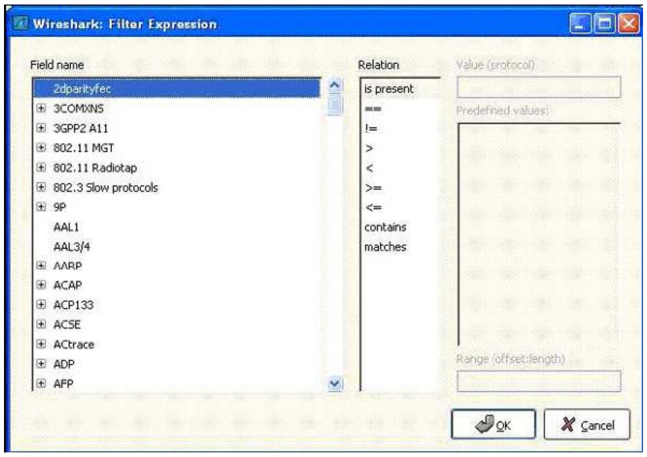
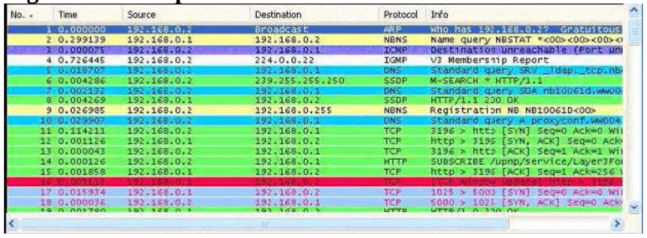
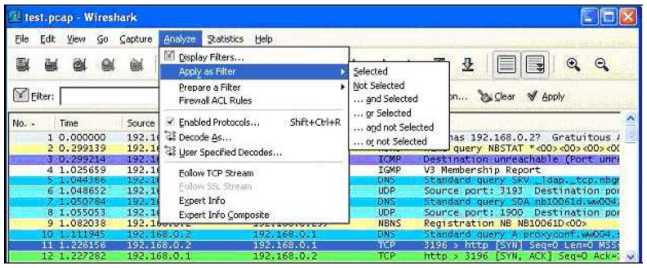


Figure 3. The expression tab



```
No. .  Time       Source          Destination       Protocol  Info
    1 0.000000   192.168.0.2     Broadcast         ARP       Who has 192.168.0.2?  Gratuitous
    2 0.299139   192.168.0.1     192.168.0.2       NBNS      Name query NBSTAT *<00><00><00><
    3 0.000075   192.168.0.2     192.168.0.1       ICMP      Destination unreachable (Port un
    4 0.726445   192.168.0.2     224.0.0.22        IGMP      V3 Membership Report
    5 0.018707   192.168.0.2     192.168.0.1       DNS       Standard query SRV _ldap._tcp.nb
    6 0.004286   192.168.0.2     239.255.255.250   SSDP      M-SEARCH * HTTP/1.1
    7 0.002132   192.168.0.2     192.168.0.1       DNS       Standard query SOA nb10061d.ww00
    8 0.004269   192.168.0.1     192.168.0.2       SSDP      HTTP/1.1 200 OK
    9 0.026985   192.168.0.2     192.168.0.255     NBNS      Registration NB NB10061D<00>
   10 0.029907   192.168.0.2     192.168.0.1       DNS       Standard query A proxyconf.ww004
   11 0.114211   192.168.0.2     192.168.0.1       TCP       3196 > http [SYN] Seq=0 Ack=0 Wi
   12 0.001126   192.168.0.1     192.168.0.2       TCP       http > 3196 [SYN, ACK] Seq=0 Ack
   13 0.000043   192.168.0.2     192.168.0.1       TCP       3196 > http [ACK] Seq=1 Ack=1 Wi
   14 0.000126   192.168.0.2     192.168.0.1       HTTP      SUBSCRIBE /upnp/service/Layer3Fo
   15 0.001858   192.168.0.1     192.168.0.2       TCP       http > 3196 [ACK] Seq=1 Ack=256
   16 0.001102   192.168.0.1     192.168.0.2       TCP       [TCP Window Update] http > 3196
   17 0.015934   192.168.0.1     192.168.0.2       TCP       1025 > 5000 [SYN] Seq=0 Ack=0 Wi
   18 0.000036   192.168.0.2     192.168.0.1       TCP       5000 > 1025 [SYN, ACK] Seq=0 Ack=
```

Figure 4. Sample Screenshot of Wireshark in action



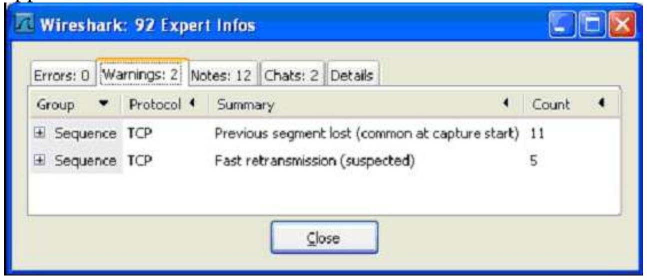
No.	Time	Source			
1	0.000000	192.16			has 192.168.0.2? Gratuitous
2	0.299139	192.16			query NBSTAT *<00><00><00><0(
3	0.299214	192.16		ICMP	Destination unreachable (Port unre
4	1.025659	192.16		IGMP	V3 Membership Report
5	1.044366	192.16		DNS	Standard query SRV _ldap._tcp.nbgr
6	1.048652	192.16		UDP	Source port: 3193 Destination por
7	1.050784	192.16		DNS	Standard query SOA nb10061d.ww004.
8	1.055053	192.16		UDP	Source port: 1900 Destination por
9	1.082038	192.16		NBNS	Registration NB NB10061D<00>
10	1.111945	192.168.0.2	192.168.0.1	DNS	Standard query A proxyconf.ww004.
11	1.226156	192.168.0.2	192.168.0.1	TCP	3196 > http [SYN] Seq=0 Len=0 MSS
12	1.227282	192.168.0.1	192.168.0.2	TCP	http > 3196 [SYN, ACK] Seq=0 Ack=

Figure 5. TCP Stream

name. Value comparisons filtering packets based on lengths secure BSSIDs source and destination addresses can be done. The menu provides tools for packet filtering and analysis precapture and during the capture itself. It eases the use of filter connectors and preparing filters. It also provides mechanisms for decoding capture files into various modes, such as the ASCII. This item can be also used to follow protocol streams of particular dialogues on particular protocols.

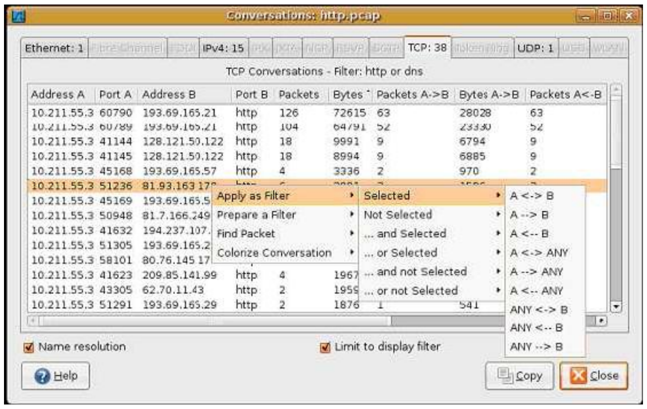
6. POST SNIFFING ANALYSIS

In this section we explore that second type of filter: display filters. The first one has already been dealt with"filtering while capturing". Display filters have applications ranging from error detection to packet sniffing and pattern identification. Worth mentioning here is a fact that Wireshark does not automatically generate alarms and alerts like a normal IDS/IPS. Instead activities that took place during a capture can be monitored and analyzed later - manually or through the use of other applications.



Group	Protocol	Summary	Count
Sequence	TCP	Previous segment lost (common at capture start)	11
Sequence	TCP	Fast retransmission (suspected)	5

Figure 6. The Information Table

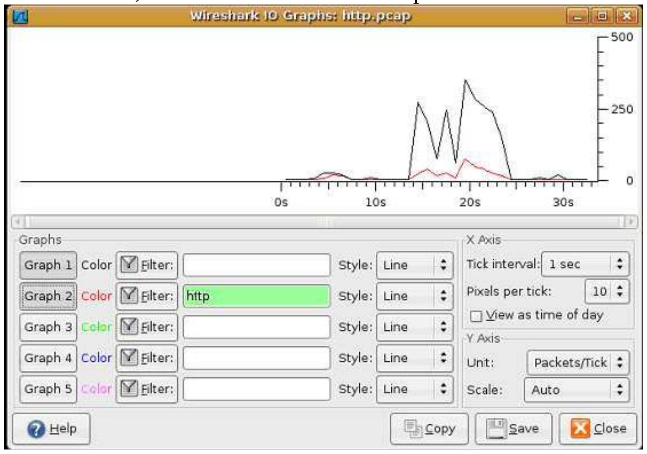


TCP Conversations - Filter: http or dns

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A->B	Bytes A->B	Packets A<-B
10.211.55.3	60790	193.69.165.21	http	126	72615	63	28028	63
10.211.55.3	60789	193.69.165.21	http	104	64791	52	23330	52
10.211.55.3	41144	128.121.50.122	http	18	9991	9	6794	9
10.211.55.3	41145	128.121.50.122	http	18	8994	9	6885	9
10.211.55.3	45168	193.69.165.57	http	4	3336	2	970	2
10.211.55.3	51236	81.93.163.17						
10.211.55.3	45169	193.69.165.5						
10.211.55.3	50948	81.7.166.249						
10.211.55.3	41632	194.237.107.						
10.211.55.3	51305	193.69.165.2						
10.211.55.3	58101	80.76.145.17						
10.211.55.3	41623	209.85.141.99	http	4	1967			
10.211.55.3	43305	62.70.11.43	http	2	1959			
10.211.55.3	51291	193.69.165.29	http	2	1876	1	541	

Figure 7. The Conversation Tool

A tool to support the mentioned arguments is the expert information table shown below (figure 6), as it visibly marks for checksum errors, redundancy checks and lost segment accounting. Another tool for intrusion and filter analysis is the menu item - statistics. Statistics of various kinds can be provided for an already captured packet, its protocol and the conversation.
It can monitor conversation of nodes passing packets between them in the captured file in the given direction. Other statistical tools are the packet summary and protocol hierarchy tools. The second major tool is the statistical IO graph (figure 8). These graphs can show flow of traffic over the network in entirety or for certain protocols only. The tool also provides the option of showing differently post filtered capture on the graph in various colors to enable easy identification, thus making Wireshark not only one of the most easily accessible sniffing software but also one of the most user friendly and comprehensible utility. Time can be set relative to the first packet or according to system clock. Usage of system clock time is effective when we are merging various capture files captured at different times. Another statistic tool worth mentioning here is the timestamp, each packet can be time stamped according to user's requirements. Even with the presence of the mentioned tools Wireshark is not by definition an IDS/IPS device although with the help of a few other utilities, like LUA and the Hex dump convertor.



Graph 1	Color	Filter:		Style: Line
Graph 2	Color	Filter:	http	Style: Line
Graph 3	Color	Filter:		Style: Line
Graph 4	Color	Filter:		Style: Line
Graph 5	Color	Filter:		Style: Line

Tick interval:	1 sec
Pixels per tick:	10
View as time of day	☐
Unit:	Packets/Tick
Scale:	Auto

Figure 8. The IO Graph tool

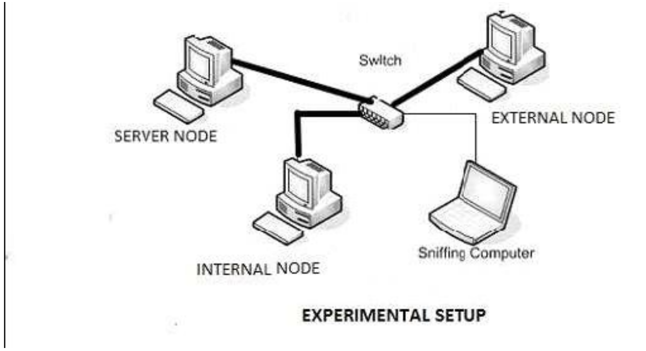


Figure 9. Experimental Setup

7. TESTING PROBLEM

The aim of the experiment below is to test the presence of unauthorized packet access to the server node i.e. the node on which unauthorized access is denied; from external node(s) i.e. experimental node, which represents a single or a group of malicious nodes in the Real Time Scenario (RTS). In the current experimental set up we have four nodes each depicting a possible node or set of nodes in a real time situation (figure 10). We have four nodes here connected by a switch (non-configurable). The nodes are as follows:

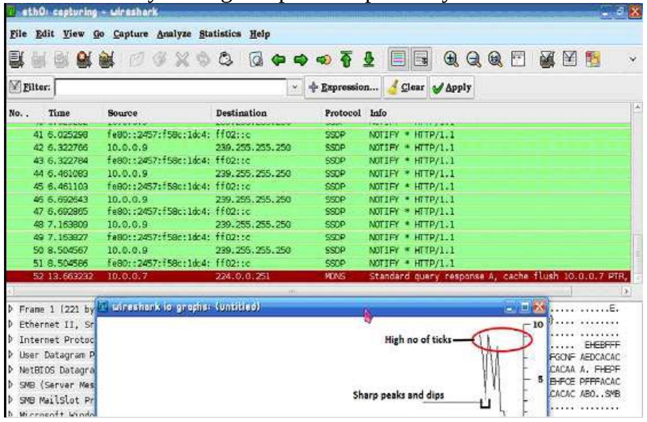
1. ANALYSER: It is the computer with Wireshark installed in it and running in promiscuous mode; IP ADDRESS: 10.0.0.12 Netmask 255.0.0.0

2. SERVER NODE: This is the node that we expect to protect from an external intrusion(although in the IDS scenario we will only be able to detect any intrusion on the server); IP ADDRESS: 10.0.0.7 Netmask 255.0.0.0

3. INTERNAL NODE: These are the nodes that can operate both on the server and connect to the outside network; IP address: 10.0.0.9 Netmask:255.0.0.0

4. EXTERNAL NODE: This is the possible intruder and under filtered mode we expect to see all possible intrusion attempts by this node on the server; IP address:10.0.0.5 Netmask 255.0.0.0

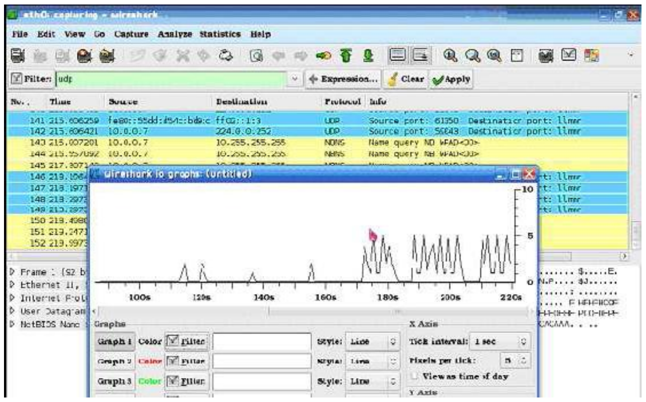
INTIAL STATE (Unfiltered Capture):
In this state no filtering expression has been used so all the traffic passing through the analyzer is being displayed here.
UDP traffic here flows from:
EXTERNAL NODE to SERVER NODE: 10.0.0.5 TO 10.0.0.7
EXTERNAL NODE to INTERNAL NODE: 10.0.0.5 TO 10.0.0.9
INTERNAL NODE to SERVER NODE : 10.0.0.9 TO 10.0.0.7
The following are the graphs obtained while observing the traffic between the above mentioned nodes during the unfiltered capture under the heavy and light capture respectively:-


eth0: capturing - wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply
wireshark io graphs: (untitled)
High no of ticks
Sharp peaks and dips
10
5

```
No. .  Time        Source                     Destination       Protocol  Info
   41 6.025298    fe80::2457:f58c:1dc4:      ff02::c           SSDP      NOTIFY * HTTP/1.1
   42 6.322766    10.0.0.9                   239.255.255.250   SSDP      NOTIFY * HTTP/1.1
   43 6.322784    fe80::2457:f58c:1dc4:      ff02::c           SSDP      NOTIFY * HTTP/1.1
   44 6.461083    10.0.0.9                   239.255.255.250   SSDP      NOTIFY * HTTP/1.1
   45 6.461103    fe80::2457:f58c:1dc4:      ff02::c           SSDP      NOTIFY * HTTP/1.1
   46 6.692643    10.0.0.9                   239.255.255.250   SSDP      NOTIFY * HTTP/1.1
   47 6.692865    fe80::2457:f58c:1dc4:      ff02::c           SSDP      NOTIFY * HTTP/1.1
   48 7.163809    10.0.0.9                   239.255.255.250   SSDP      NOTIFY * HTTP/1.1
   49 7.163827    fe80::2457:f58c:1dc4:      ff02::c           SSDP      NOTIFY * HTTP/1.1
   50 8.504567    10.0.0.9                   239.255.255.250   SSDP      NOTIFY * HTTP/1.1
   51 8.504586    fe80::2457:f58c:1dc4:      ff02::c           SSDP      NOTIFY * HTTP/1.1
   52 13.663232   10.0.0.7                   224.0.0.251       MDNS      Standard query response A, cache flush 10.0.0.7 PTR,
```

Figure 10. Graph Showing Short Capture During Heavy Traffic Flow

The presence of sharp peaks is due to UDP traffic flow.


eth0: capturing - wireshark
File Edit View Go Capture Analyze Statistics Help
Filter: udp
Expression... Clear Apply
wireshark io graphs: (untitled)
10
5
0
100s
120s
140s
160s
180s
200s
220s
Graphs
Graph 1 Color Filter Style: Line
Graph 2 Color Filter Style: Line
Graph 3 Color Filter Style: Line
X Axis
Tick interval: 1 sec
Pixels per tick: 5
View as time of day
Y Axis

```
No. .  Time         Source                   Destination       Protocol  Info
  141 215.606259   fe80::55dd:f54c::bdc:c   ff02::1:3         UDP       Source port: 61350  Destination port: llmnr
  142 215.606421   10.0.0.7                 224.0.0.252       UDP       Source port: 56843  Destination port: llmnr
  143 215.607201   10.0.0.7                 10.255.255.255    NBNS      Name query NB WPAD<00>
  144 215.357092   10.0.0.7                 10.255.255.255    NBNS      Name query NB WPAD<00>
```

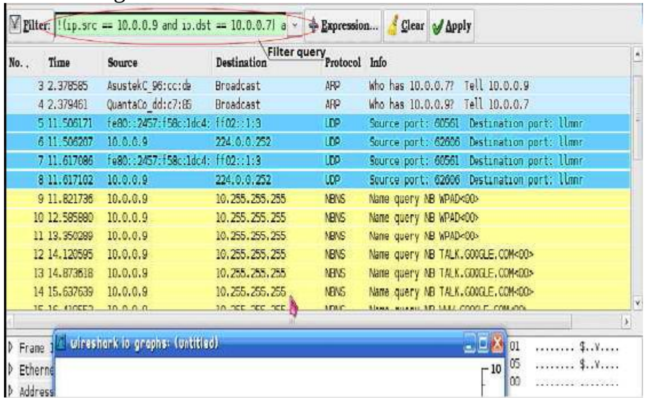
Figure 11. Graph Showing Captured Traffic during Experimental Setup

The peaks (ticks) in this graph are lower compared to the previous graphs because of lower traffic flow as the lesser data flow.

FINAL SET UP:
Filter queries used:
1) IP.SRC == 10.0.0.5 and IP.DST == 10.0.0.9
2) IP.SRC == 10.0.0.9 and IP.DST == 10.0.0.7
3) UDP

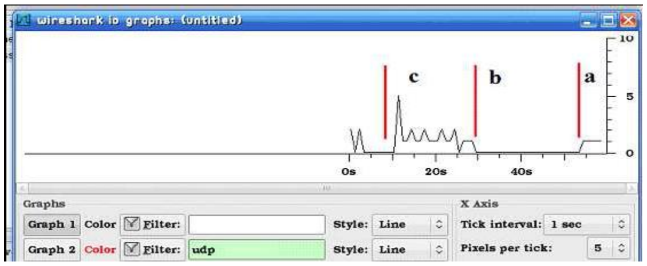
Operators used:
Logical AND :- and
Logical Negation :- !
Logical OR: - or
Final Query: - udp and !(ip.src == 10.0.0.5 and ip.dst == 10.0.0.9) and (ip.src == 10.0.0.9 and ip.dst == 10.0.0.7)

As a result this query would filter all the traffic from the external node to the servers and will show that on the captured file rest all data packets will go unmonitored. In case of unwanted access to the server, the external node will be marked visible. Wireshark alone will not be able to generate an alarm or take a security action against the unauthorized access, it can only maintain track of unauthorized accesses but with use of other utilities, alert generation is also possible. For better understanding and comprehensibility, we show a graph depicting the filtered flow as shown in figure 12:


Filter: !(ip.src == 10.0.0.9 and ip.dst == 10.0.0.7) a
Expression... Clear Apply
Filter query
wireshark io graphs: (untitled)
10

```
No. .  Time        Source                     Destination       Protocol  Info
    3 2.378585    AsustekC_96:cc:de          Broadcast         ARP       Who has 10.0.0.7?  Tell 10.0.0.9
    4 2.379461    QuantaCo_dd:c7:86          Broadcast         ARP       Who has 10.0.0.9?  Tell 10.0.0.7
    5 11.506171   fe80::2457:f58c:1dc4:      ff02::1:3         UDP       Source port: 60561  Destination port: llmnr
    6 11.506207   10.0.0.9                   224.0.0.252       UDP       Source port: 62606  Destination port: llmnr
    7 11.617086   fe80::2457:f58c:1dc4:      ff02::1:3         UDP       Source port: 60561  Destination port: llmnr
    8 11.617102   10.0.0.9                   224.0.0.252       UDP       Source port: 62606  Destination port: llmnr
    9 11.821736   10.0.0.9                   10.255.255.255    NBNS      Name query NB WPAD<00>
   10 12.585880   10.0.0.9                   10.255.255.255    NBNS      Name query NB WPAD<00>
   11 13.350289   10.0.0.9                   10.255.255.255    NBNS      Name query NB WPAD<00>
   12 14.120595   10.0.0.9                   10.255.255.255    NBNS      Name query NB TALK.GOOGLE.COM<00>
   13 14.873618   10.0.0.9                   10.255.255.255    NBNS      Name query NB TALK.GOOGLE.COM<00>
   14 15.637639   10.0.0.9                   10.255.255.255    NBNS      Name query NB TALK.GOOGLE.COM<00>
```

Figure 12. Graph Depicting Filtered Flow



Graph 1 Color Filter: Style: Line
Graph 2 Color Filter: udp Style: Line
Tick interval: 1 sec
Pixels per tick: 5

Figure 13. Graph Showing 3 Regions

The regions in the graph above (figure 13) are explained as below:-

a. Region a: This is the region of beacon and control traffic flow at the initiation of network and shows no sharp peaks.

b. Region b: Activity in the network is from the internal node to the server and between external and internal node. So, capture does not show any data at all.

c. Region c: Malicious activity starts at this point of time and is accompanied by UDP activity in the packet capture pane and the sharp peaks in the I/O graph.

8. OBSERVATIONS

The experiment shows two scenarios, in the first scenario Wireshark captures all traffic, this shows peaks of varying heights in the IO graph. This is the unguarded mode where wireshark simply monitors activities and notifies users of the basic errors in packet traversal. The second scenario makes use of filtered capture on the basis of filters as mentioned above (capture will not be shown for any ordinary activity) and we see that IO graph shows no activity for a certain time. Though, as soon as the malicious node becomes active and starts sending data to the critical server node capture begins and activity is shown in the IO graph.

9. CONCLUSION

The above experiment asserts the need of IDS/IPS devices in any typical network. We have also highlighted the capabilities of Wireshark in packet data interpretation and data handling too. Wireshark, in this experiment has been used primarily in ACL (Access Control List) filtering. Many other variations of filtering are available in the Wireshark utility such as filtering based on packet size, filtering based on protocols used, filtering of sub-strings etc. Thus, with proper use of filtering commands and complementing utilities, Wireshark can be developed into comprehensive intrusion detection software.

10. FUTURE WORK

Wireshark as a Network Protocol Analyzer has already proven its mettle in all necessary realms. However it still has scope of improvement in it as far as alert generation and heuristic development is concerned. We are working to introduce certain utilities in the source code of Wireshark to overcome the above shortcomings by making Wireshark capable of alert generations.

11. ACKNOWLEDGMENTS

This work is part of a WOS-A project (ref. no. : SR/WOSA/ET-20/2008) funded by the Department of Science and Technology, Government of India.

12. REFERENCES

[1] Roesch M (1999) Snort - Lightweight Intrusion Detection for Networks. In Proceedings of Thirteenth Systems Administration Conference (LISA), pp 229-238.

[2] Stolze M, Pawlitzek R and Hild S (2003a) Task Support for Network Security Monitoring. In ACM CHI Workshop on System Administrators Are Users, Too: Designing Workspaces for Managing Internet-Scale Systems.

[3] Lee W, Stolfo SJ and Mok KW (2000) Adaptive Intrusion Detection: A Data Mining Approach. Artificial Intelligence Review 14(6), 533-567.

[4] Stolze M, Pawlitzek R and Wespi A (2003b) Visual Problem-Solving Support for New Event Triage in Centralized Network Security Monitoring: Challenges, Tools and Benefits. In GI-SIDAR conference IT-Incident Management and IT-Forensics (IMF).

[5] Pinkas, B., Sander, T.: Securing passwords against dictionary attacks Proceedings of the 9th ACM conference on Computer and communications security Washington, DC, USA (2002) 161-170

[6] Madsen, P., Koga, Y., Takahashi, K.: Federated identity management for protecting users from ID theft Proceedings of the 2005 workshop on Digital identity management Fairfax, VA, USA (2005) 77-83

[7] Gouda, M.G., Liu, A.X., Leung, L.M., Alam, M.A.: Single Password, Multiple Accounts. Proceedings of 3rd Applied Cryptography and Network Security Conference (industry track), New York City, New York (2005)

[8] Luo, H., Henry, P.: A common password method for protection of multiple accounts. 14th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, Vol. 3 (2003) 2749 - 2754

[9] Gaw, S., Felten, E.W.: Password management strategies for online accounts. Proceedings of the second symposium on Usable privacy and security ACM Press, Pittsburgh, Pennsylvania (2006) 44-55

[10] Riley, S.: Password Security: What Users Know and What They Actually Do. Usability News, Vol. 2006. Software Usability Research Laboratory, Department of Psychology, Wichita State University, Wichita (2006)

[11] S. R. Snapp, J. Brentano, G. V. Dias, T. L. Goan, L. T. Heberlein, C. Ho, K. N. Levitt, B. Mukherjee, S. E. Smaha, T. Grance, D. M. Teal and D. Mansur, DIDS (Distributed Intrusion Detection System) Motivation, Architecture and Early Prototype, Proceeding 14th National Computer Security Conference, pg. 167 176, 1991

[12] S. James P. Anderson, Computer security threat monitoring and surveillance" ,Technical report, Fort Washington, PA, April 1980

[13] Stephen E. Smaha, "Haystack: An intrusion detection system", In Proceedings of the Fourth Aerospace Computer Security Applications Conference, pages 37-44, December 1988.

[14] L. Todd Heberlein, Gihan V. Dias, Karl N. Levitt, Biswanath Mukherjee, Jeff Wood, and David Wolber, "A network security monitor", In Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy, pages 296-304, May 1990.