Network Forensics using Snort and Wireshark

Sakshi Singh*[1],Suresh Kumar[2]

[1,2]Ambedkar Institute of Advanced Communication Technologies and Research, Delhi, India
[1]010sakshisingh@gmail.com,[3]sureshpoonia@aiactr.ac.in

Abstract

The data and network security have been anissue of primary concern for most of the organizations as dependence on the exchange of information has increased exponentially. Network monitoring has now become an essential information protection aspect of defending attacks. Intrusion detection system (IDS) plays a vital role in network monitoring tools such as Wireshark and Snort. Wireshark and Snort can monitor intrusive electronic information graphically on network processes or movements.Network management using IDS and intrusion prevention system (IPS) improves network connectivity, efficiency and security. The Snort-IDS/IPS intrusion detection and prevention system is an open-source network security tool that has been used extensively to protect the organization's network. In this article, we captured network traffic in real-time using Snort, and analyzed those captured packets by Wireshark for a detailed network analysis report. We also examined these packets in real-time.

Keywords: *Data packets, Packet analysis, Snort, Wireshark*

1. Introduction

Our internet dependency has significantly increased. With that internet usage, various security threats have emerged, which placed organizations and individuals at risk for secrecy, credibility, and access to online data. There are other strategies for combating these attacks, including anti-viruses, firewalls, and IDS. While all techniques couldbe used at the host level, network intrusion detection systems have been designed to monitor activities and detect risks infesting individual network objectives. Network intrusion detection systems are specialized IDS.

Snort is the open-source IDS framework for intrusion prevention,analyzes traffic and monitor packets on IP networks. For the data packets traffic, the Snort-IDS uses the rules. Snort-IDS generates alert messages if a packet matches those rules. It can analyze protocols, search and fit the content and can be used for a variety of attacks and samples. It sniffs the network packets and matches every received packet with its rule collection to detect any maliciousbehavior.

The statistical tools of Wireshark are one of the strengths. To create statistics and graphs, the user may use network traffic knowledge. To crack down the authenticated files, users can use the Decryption Techniques of the application. Wireshark has a default three-pane packet explorer that can support the caught packet in any detail. In this job, we define the violation by finding the real network traffic with Snort and examine the captured packet in depth using the Wireshark network monitoring method. We monitor actual traffic from the wired or wireless media and track intrusions on snort. Snort uses the predefined law to track a network state. It can provide a compact packet detail. When the packet just meets the requirement, it is registered, and packets dropped otherwise. For the thorough review of the selected packet, these recorded files have been imported into the Wireshark interface.

2. Literature Review

Author [1] implemented a network intrusion system using an anomaly tool by considering machine learning. This method recorded the live network transmission and used Wireshark and snort for thorough study. Wireshark developed the I/O graph of caught network packets that shows the network dynamics information and an overview of the problems.

In [2] author presented a method that uses Wireshark in the analysis of network protocols. He used analysis to detect conventional network threats, such as port search, hidden FTP and IRC networks, ICMP-based attacks, Bit-Torrent denial services, etc. He has also demonstrated that packet inspection can identify a wide variety of security vulnerabilities and assaults on networked computer networks.

In [3]author proposed a framework to avoid distributed denial of service (DDOS) attacks in cloud computing employing Snort IDS / IPS and Wireshark. Further, author clarified the services available on-demand in cloud storage to allow the consumer to access such services through the internet. Denial of service (DOS) and DDOS are the most significant attacks which take advantage of cloud computing vulnerabilities. The author offered an experimental design to prevent DDOS attacks but factors such as consistency and expense were not considered.

In[4] author showed that the network protocol analysis is a sniffer to collect data for analysis and technical understanding of packets. A comprehensive study of network attacks with protection, network sniffer, and protocol analyzer was presented.

In[5] author proposed method to test the efficiency and identification accuracy of the distributed denial of service attack in terms of TCP flooding. The paper examined significant factors that influence the efficiency and detection abilities of snort and proposed strategies for improving the IDS. The studies have shown that improved equipment can use to strengthen snort's handling efficiency.

3. Snort as NIDS

Snort is a Sourcefire built open-source network intrusion and sensing program (IDS / IPS). Snort is the most widely used IDS / IPS technology in the world and brings together the advantages of signatures, protocols, and specifications based on exceptions. The Snort is an avoidance and identification IDPS (Intrusion Detection and Prevention System) thatprimarily study concepts related to identification of signature and irregularities [1].Snort compares network path signatures with preset signatures in the Snorts library or database that can be continually modified while signature-based identification produced. If the pattern is matched with the detected signature, it improves output, but no new attack styles are identified that do not exists in the snort database.

Snort will undertake traffic analysis and packet logging in real-time on Internet Protocol (IP) networks. Snort analyzes, checks, and matches for protocols. The software may also identify samples or threats, but not just fingerprinting attempts, symbolic URL assaults, buffer overflows, application message chain tests, covert port scans [3].

4. Analysis Capabilities of Wireshark

Wireshark is a packet analyzer for the network. A network packet analyzer tends to be accurate with captured packet details. In real-time, this program analyzes network traffic. Wireshark has the I/O graph that sums up the packet's transit. Drilling I/O Graphs show the total traffic, generally measured in bytes or packets per second, on the capture file [6]. The network communication is the link between two different endpoints. The communication window consists of addresses, packet and byte counters, beginning time and duration of conversation along with bits/sec in each direction. There are specialist details on the malformed packet and the extent of the packet and packet number in Wireshark. The identification in which IP to connect may be performed in a precise manner, for example, graphical analysis [7].

5. Methodology

The methodology defines the process used to get the final analysis report of the captured live data packets. A snort from the internet will achieve live data packets. A snort is an open-source tool that matches predefined signatures with the captured packets and sends alert messages to the user if any match found. In our methodology, we used Ethernet cable to get access to the internet so that we can get an accurate flow of data packets. After initializing the capturing process, Snort creates a log file in its directory in which it stores all the captured packets. The log file consists of all the data packets with their alert messages. A log file is created by Snort that stores all the captured packets with their alert messages. The log file is exported to Wireshark to analyze the captured network packets. Wireshark generates every single detail about the packets of the log file.

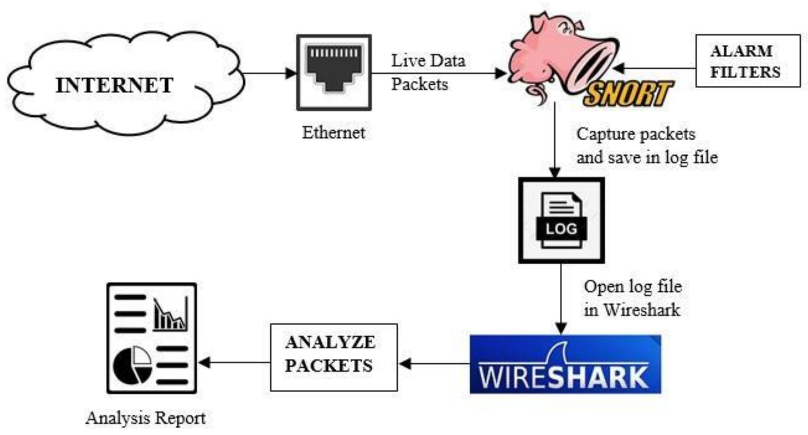


Figure 1. Architecture of network analysis process using Snort and Wireshark

6. Experiment

6.1 Snort Configuration

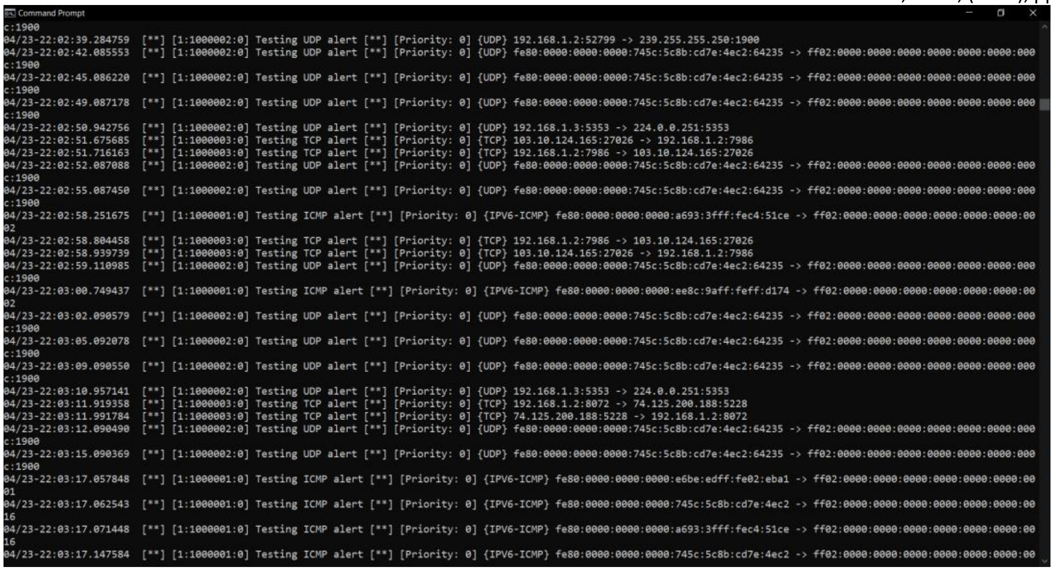
We first visit < http:/www.snort.org/ > to configure Snort and only get to use Snort and the guidelines. Snort Configuration: We need Winpcap tools to store packages to run snort on Windows. By using C:\Snort etc, get the snort.conf file and open this WordPad file, we took all the measures needed to create our custom configurations.

Table1. Snort Configuration

Set Network Variables	Network Address External Network Address The path for Snort Rules Path for Preproc Rules	HOME_NET 192.168.1.0/24 EXTERNAL_NET !$HOME.NET c:\Snort\rules WHITE_LIST_PATH c:\Snort\rules BLACK_LIST_PATH c:\Snort\rules c:\Snort\preproc_rules
Configure the Decoder	Config logdir	c:\Snort\log
Configure Dynamic loaded Libraries	Path to dynamic preprocessor libraries Path to base preprocessor engine	c:\Snort\lib\snort_dynamicpreprocesso\ c:\Snort\lib\snort_dynamicengine\sf_engine.dll
Configure Preprocessor	Reputation preprocessor	$WHITE_LIST_PATH\White.list $BLACK_LIST_PATH\Black.list
Classification	-	c:\Snort\etc\snort.conf
Local Rules	Local.rules	alert HTTP any any -> any any (msg: “Testing HTTP alert”; sid: 1000001;) alert UDP any any -> any any (msg: “Testing UDP alert”; sid: 1000002;) alert SSDP any any -> any any (msg: “Testing SSDP alert”; sid: 1000003;)

For configurations in the configuration file, we have to snort the snort.conf file address.
snort -v -c \etc\snort\snort.conf

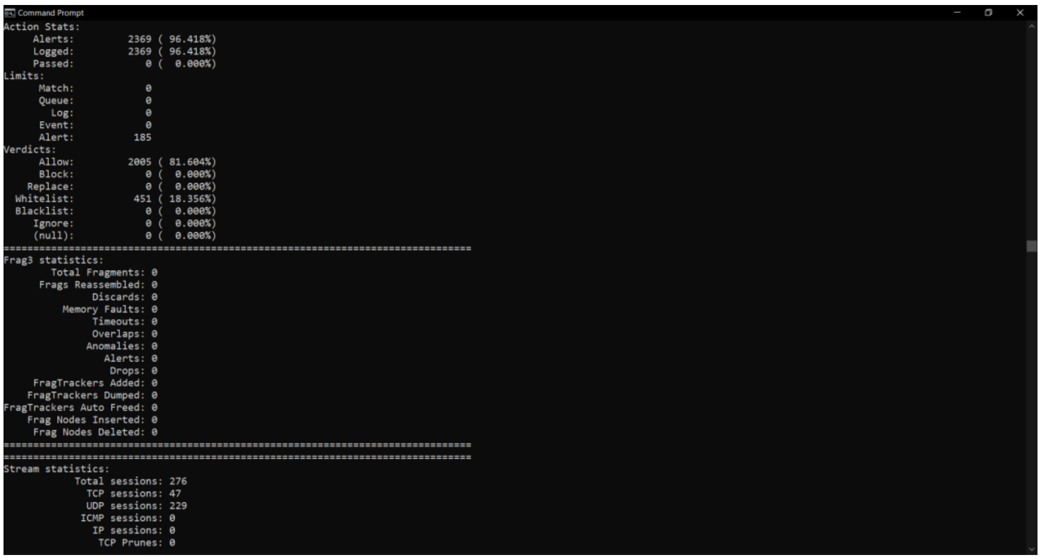
The following command used to collect data packets from the network:
Snort –i 6 –c c:\snort\etc\snort.conf –A console



```
c:1900
04/23-22:02:39.284759  [**] [1:1000002:0] Testing UDP alert [**] [Priority: 0] {UDP} 192.168.1.2:52799 -> 239.255.255.250:1900
04/23-22:02:42.085553  [**] [1:1000002:0] Testing UDP alert [**] [Priority: 0] {UDP} fe80:0000:0000:0000:745c:5c8b:cd7e:4ec2:64235 -> ff02:0000:0000:0000:0000:0000:000
c:1900
04/23-22:02:45.086220  [**] [1:1000002:0] Testing UDP alert [**] [Priority: 0] {UDP} fe80:0000:0000:0000:745c:5c8b:cd7e:4ec2:64235 -> ff02:0000:0000:0000:0000:0000:000
c:1900
04/23-22:02:49.087178  [**] [1:1000002:0] Testing UDP alert [**] [Priority: 0] {UDP} fe80:0000:0000:0000:745c:5c8b:cd7e:4ec2:64235 -> ff02:0000:0000:0000:0000:0000:000
c:1900
04/23-22:02:50.942756  [**] [1:1000002:0] Testing UDP alert [**] [Priority: 0] {UDP} 192.168.1.3:5353 -> 224.0.0.251:5353
04/23-22:02:51.675685  [**] [1:1000003:0] Testing TCP alert [**] [Priority: 0] {TCP} 103.10.124.165:27026 -> 192.168.1.2:7986
04/23-22:02:51.716163  [**] [1:1000003:0] Testing TCP alert [**] [Priority: 0] {TCP} 192.168.1.2:7986 -> 103.10.124.165:27026
04/23-22:02:52.087088  [**] [1:1000002:0] Testing UDP alert [**] [Priority: 0] {UDP} fe80:0000:0000:0000:745c:5c8b:cd7e:4ec2:64235 -> ff02:0000:0000:0000:0000:0000:000
c:1900
04/23-22:02:55.087450  [**] [1:1000002:0] Testing UDP alert [**] [Priority: 0] {UDP} fe80:0000:0000:0000:745c:5c8b:cd7e:4ec2:64235 -> ff02:0000:0000:0000:0000:0000:000
c:1900
04/23-22:02:58.251675  [**] [1:1000001:0] Testing ICMP alert [**] [Priority: 0] {IPV6-ICMP} fe80:0000:0000:0000:a693:3fff:fec4:51ce -> ff02:0000:0000:0000:0000:0000:0000:00
02
04/23-22:02:58.804458  [**] [1:1000003:0] Testing TCP alert [**] [Priority: 0] {TCP} 192.168.1.2:7986 -> 103.10.124.165:27026
04/23-22:02:58.939739  [**] [1:1000003:0] Testing TCP alert [**] [Priority: 0] {TCP} 103.10.124.165:27026 -> 192.168.1.2:7986
04/23-22:02:59.110985  [**] [1:1000002:0] Testing UDP alert [**] [Priority: 0] {UDP} fe80:0000:0000:0000:745c:5c8b:cd7e:4ec2:64235 -> ff02:0000:0000:0000:0000:0000:000
c:1900
04/23-22:03:00.749437  [**] [1:1000001:0] Testing ICMP alert [**] [Priority: 0] {IPV6-ICMP} fe80:0000:0000:0000:ee8c:9aff:feff:d174 -> ff02:0000:0000:0000:0000:0000:0000:00
02
04/23-22:03:02.090579  [**] [1:1000002:0] Testing UDP alert [**] [Priority: 0] {UDP} fe80:0000:0000:0000:745c:5c8b:cd7e:4ec2:64235 -> ff02:0000:0000:0000:0000:0000:000
c:1900
04/23-22:03:05.092078  [**] [1:1000002:0] Testing UDP alert [**] [Priority: 0] {UDP} fe80:0000:0000:0000:745c:5c8b:cd7e:4ec2:64235 -> ff02:0000:0000:0000:0000:0000:000
c:1900
04/23-22:03:09.090550  [**] [1:1000002:0] Testing UDP alert [**] [Priority: 0] {UDP} fe80:0000:0000:0000:745c:5c8b:cd7e:4ec2:64235 -> ff02:0000:0000:0000:0000:0000:000
c:1900
04/23-22:03:10.957141  [**] [1:1000002:0] Testing UDP alert [**] [Priority: 0] {UDP} 192.168.1.3:5353 -> 224.0.0.251:5353
04/23-22:03:11.919358  [**] [1:1000003:0] Testing TCP alert [**] [Priority: 0] {TCP} 192.168.1.2:8072 -> 74.125.200.188:5228
04/23-22:03:11.991784  [**] [1:1000003:0] Testing TCP alert [**] [Priority: 0] {TCP} 74.125.200.188:5228 -> 192.168.1.2:8072
04/23-22:03:12.090490  [**] [1:1000002:0] Testing UDP alert [**] [Priority: 0] {UDP} fe80:0000:0000:0000:745c:5c8b:cd7e:4ec2:64235 -> ff02:0000:0000:0000:0000:0000:000
c:1900
04/23-22:03:15.090369  [**] [1:1000002:0] Testing UDP alert [**] [Priority: 0] {UDP} fe80:0000:0000:0000:745c:5c8b:cd7e:4ec2:64235 -> ff02:0000:0000:0000:0000:0000:000
c:1900
04/23-22:03:17.057848  [**] [1:1000001:0] Testing ICMP alert [**] [Priority: 0] {IPV6-ICMP} fe80:0000:0000:0000:e6be:edff:fe02:eba1 -> ff02:0000:0000:0000:0000:0000:0000:00
01
04/23-22:03:17.062543  [**] [1:1000001:0] Testing ICMP alert [**] [Priority: 0] {IPV6-ICMP} fe80:0000:0000:0000:745c:5c8b:cd7e:4ec2 -> ff02:0000:0000:0000:0000:0000:0000:00
16
04/23-22:03:17.071448  [**] [1:1000001:0] Testing ICMP alert [**] [Priority: 0] {IPV6-ICMP} fe80:0000:0000:0000:a693:3fff:fec4:51ce -> ff02:0000:0000:0000:0000:0000:0000:00
16
04/23-22:03:17.147584  [**] [1:1000001:0] Testing ICMP alert [**] [Priority: 0] {IPV6-ICMP} fe80:0000:0000:0000:745c:5c8b:cd7e:4ec2 -> ff02:0000:0000:0000:0000:0000:0000:00
```

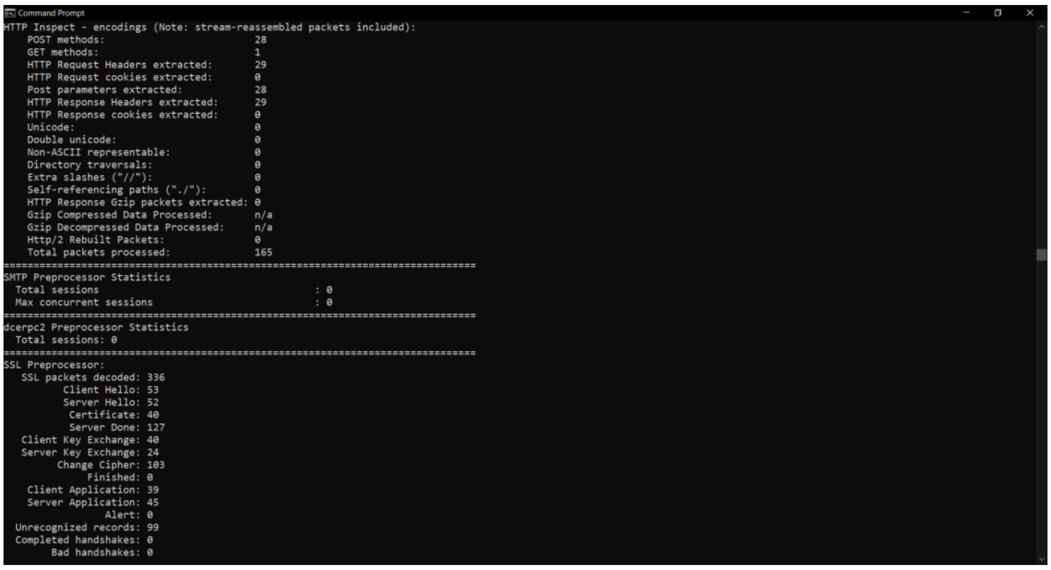
Figure 2. Live data packets captured by Snort with their alert messages

Ctrl+C stops the packet capturing process, and provides a summary.



```
Action Stats:
     Alerts:         2369 ( 96.418%)
     Logged:         2369 ( 96.418%)
     Passed:            0 (  0.000%)
Limits:
      Match:            0
      Queue:            0
        Log:            0
      Event:            0
      Alert:          185
Verdicts:
      Allow:         2005 ( 81.604%)
      Block:            0 (  0.000%)
    Replace:            0 (  0.000%)
  Whitelist:          451 ( 18.356%)
  Blacklist:            0 (  0.000%)
     Ignore:            0 (  0.000%)
     (null):            0 (  0.000%)
===============================================================================
Frag3 statistics:
        Total Fragments: 0
      Frags Reassembled: 0
               Discards: 0
          Memory Faults: 0
               Timeouts: 0
               Overlaps: 0
              Anomalies: 0
                 Alerts: 0
                  Drops: 0
     FragTrackers Added: 0
    FragTrackers Dumped: 0
FragTrackers Auto Freed: 0
    Frag Nodes Inserted: 0
     Frag Nodes Deleted: 0
===============================================================================
===============================================================================
Stream statistics:
            Total sessions: 276
              TCP sessions: 47
              UDP sessions: 229
             ICMP sessions: 0
               IP sessions: 0
                TCP Prunes: 0
```

Figure 3. Summary of captured data packets by Snort- NIDS mode - I



```
HTTP Inspect - encodings (Note: stream-reassembled packets included):
    POST methods:                         28
    GET methods:                          1
    HTTP Request Headers extracted:       29
    HTTP Request cookies extracted:       0
    Post parameters extracted:            28
    HTTP Response Headers extracted:      29
    HTTP Response cookies extracted:      0
    Unicode:                              0
    Double unicode:                       0
    Non-ASCII representable:              0
    Directory traversals:                 0
    Extra slashes ("//"):                 0
    Self-referencing paths ("./"):        0
    HTTP Response Gzip packets extracted: 0
    Gzip Compressed Data Processed:       n/a
    Gzip Decompressed Data Processed:     n/a
    Http/2 Rebuilt Packets:               0
    Total packets processed:              165
===============================================================================
SMTP Preprocessor Statistics
  Total sessions                                    : 0
  Max concurrent sessions                           : 0
===============================================================================
dcerpc2 Preprocessor Statistics
  Total sessions: 0
===============================================================================
SSL Preprocessor:
   SSL packets decoded: 336
          Client Hello: 53
          Server Hello: 52
           Certificate: 40
           Server Done: 127
   Client Key Exchange: 40
   Server Key Exchange: 24
         Change Cipher: 103
              Finished: 0
    Client Application: 39
    Server Application: 45
                 Alert: 0
  Unrecognized records: 99
  Completed handshakes: 0
        Bad handshakes: 0
```

Figure 4. Summary of captured data packets by Snort- NIDS mode - II

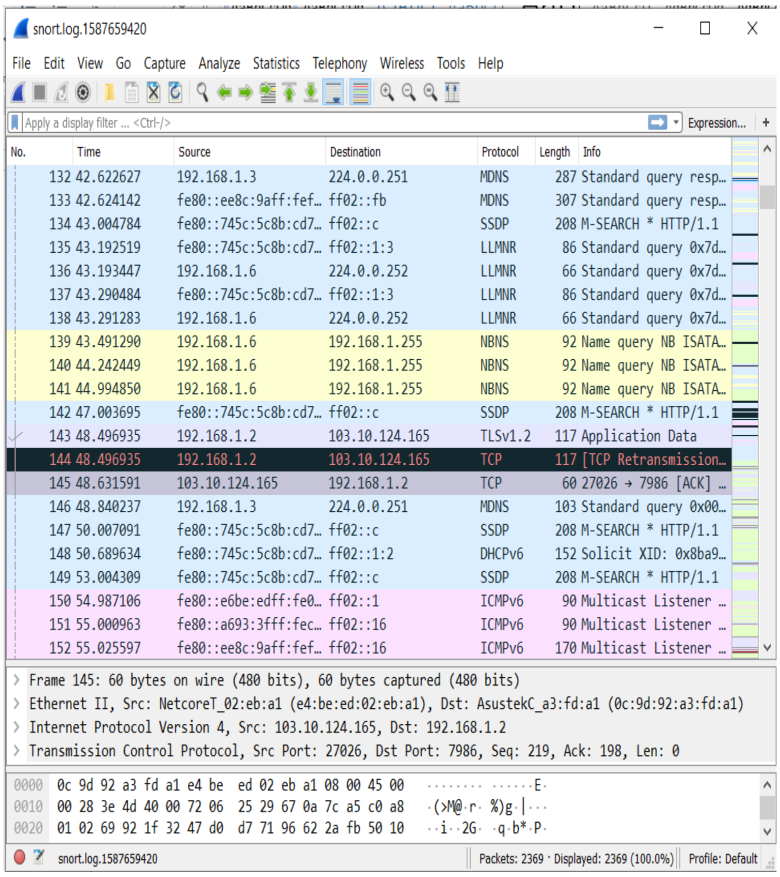
The collected data is stored for each IP address in packet logging mode in a directory hierarchy of subdirectories. The file structure for collected information is now used by regular snort tcpdump (pcap) so that all data gathered go into one server. Utilizing the "-K ASCII" method, the client uses the remote or localhost address to build subdirectory and log packets [8].

Snort logs the packets from the host to host that flow from both. Snort uses the remote address of the device as the directory where packets are stored and uses localhost address when defining a basic -l-option. In conjunction with the home network, we must inform Snort which network is the house (local end) network and should use the "h net/mask" option to log into remote folders with all traffic captured:

```
snort -de -K ASCII -l \tmp\snort-log \
      -h 192.168.1.0/24 -c \etc\snort\snort.conf
```

6.2 Export to Wireshark

For a detailed analysis of the captured packet, the produced log file is exported to Wireshark. Here the study of the different protocols is performed. If the file is accessed, all the data gathered can be viewed. It displays the packet source and target address, the protocol, and the packet information. The figure below indicates the network traffic reported and imported from the Wireshark interface by the snort. Through picking a package, one can see the specifics of each package.


snort.log.1587659420
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...

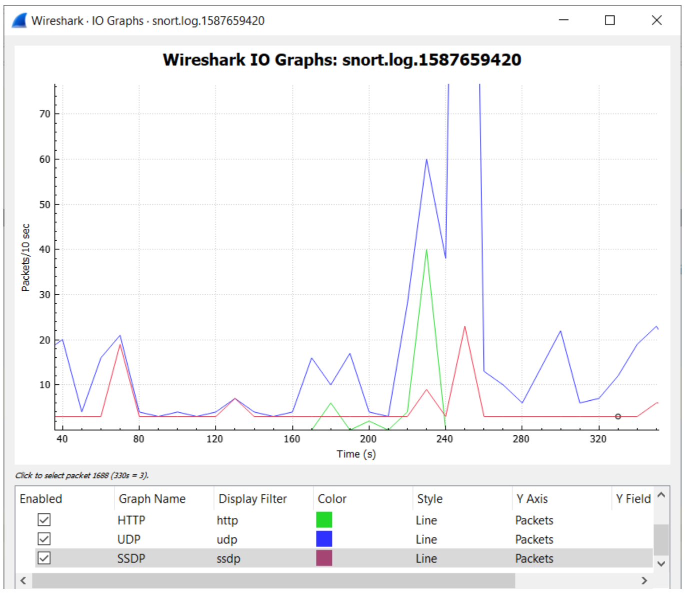
```
No.  Time       Source               Destination    Protocol Length Info
 132 42.622627  192.168.1.3          224.0.0.251    MDNS     287 Standard query resp…
 133 42.624142  fe80::ee8c:9aff:fef… ff02::fb       MDNS     307 Standard query resp…
 134 43.004784  fe80::745c:5c8b:cd7… ff02::c        SSDP     208 M-SEARCH * HTTP/1.1
 135 43.192519  fe80::745c:5c8b:cd7… ff02::1:3      LLMNR     86 Standard query 0x7d…
 136 43.193447  192.168.1.6          224.0.0.252    LLMNR     66 Standard query 0x7d…
 137 43.290484  fe80::745c:5c8b:cd7… ff02::1:3      LLMNR     86 Standard query 0x7d…
 138 43.291283  192.168.1.6          224.0.0.252    LLMNR     66 Standard query 0x7d…
 139 43.491290  192.168.1.6          192.168.1.255  NBNS      92 Name query NB ISATA…
 140 44.242449  192.168.1.6          192.168.1.255  NBNS      92 Name query NB ISATA…
 141 44.994850  192.168.1.6          192.168.1.255  NBNS      92 Name query NB ISATA…
 142 47.003695  fe80::745c:5c8b:cd7… ff02::c        SSDP     208 M-SEARCH * HTTP/1.1
 143 48.496935  192.168.1.2          103.10.124.165 TLSv1.2  117 Application Data
 144 48.496935  192.168.1.2          103.10.124.165 TCP      117 [TCP Retransmission…
 145 48.631591  103.10.124.165       192.168.1.2    TCP       60 27026 → 7986 [ACK] …
 146 48.840237  192.168.1.3          224.0.0.251    MDNS     103 Standard query 0x00…
 147 50.007091  fe80::745c:5c8b:cd7… ff02::c        SSDP     208 M-SEARCH * HTTP/1.1
 148 50.689634  fe80::745c:5c8b:cd7… ff02::1:2      DHCPv6   152 Solicit XID: 0x8ba9…
 149 53.004309  fe80::745c:5c8b:cd7… ff02::c        SSDP     208 M-SEARCH * HTTP/1.1
 150 54.987106  fe80::e6be:edff:fe0… ff02::1        ICMPv6    90 Multicast Listener …
 151 55.000963  fe80::a693:3fff:fec… ff02::16       ICMPv6    90 Multicast Listener …
 152 55.025597  fe80::ee8c:9aff:fef… ff02::16       ICMPv6   170 Multicast Listener …

> Frame 145: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
> Ethernet II, Src: NetcoreT_02:eb:a1 (e4:be:ed:02:eb:a1), Dst: AsustekC_a3:fd:a1 (0c:9d:92:a3:fd:a1)
> Internet Protocol Version 4, Src: 103.10.124.165, Dst: 192.168.1.2
> Transmission Control Protocol, Src Port: 27026, Dst Port: 7986, Seq: 219, Ack: 198, Len: 0

0000  0c 9d 92 a3 fd a1 e4 be  ed 02 eb a1 08 00 45 00   ········ ······E·
0010  00 28 3e 4d 40 00 72 06  25 29 67 0a 7c a5 c0 a8   ·(>M@·r· %)g·|···
0020  01 02 69 92 1f 32 47 d0  d7 71 96 62 2a fb 50 10   ··i··2G· ·q·b*·P·

snort.log.1587659420    Packets: 2369 · Displayed: 2369 (100.0%)    Profile: Default
```

Figure 5. Data packet analysis by Wireshark

7. Result

Wireshark is a flexible network packet analyzer method for collecting and reliably detailed packets streaming through the online network. It displays every aspect of the packet graphically as shown in Fig 5. The Header component consists of IP address, protocol, live field date, variant of the protocol, header duration, and specific categories of services. The header data is displayed in the decimal format, while payload data presented in the hexadecimal form.

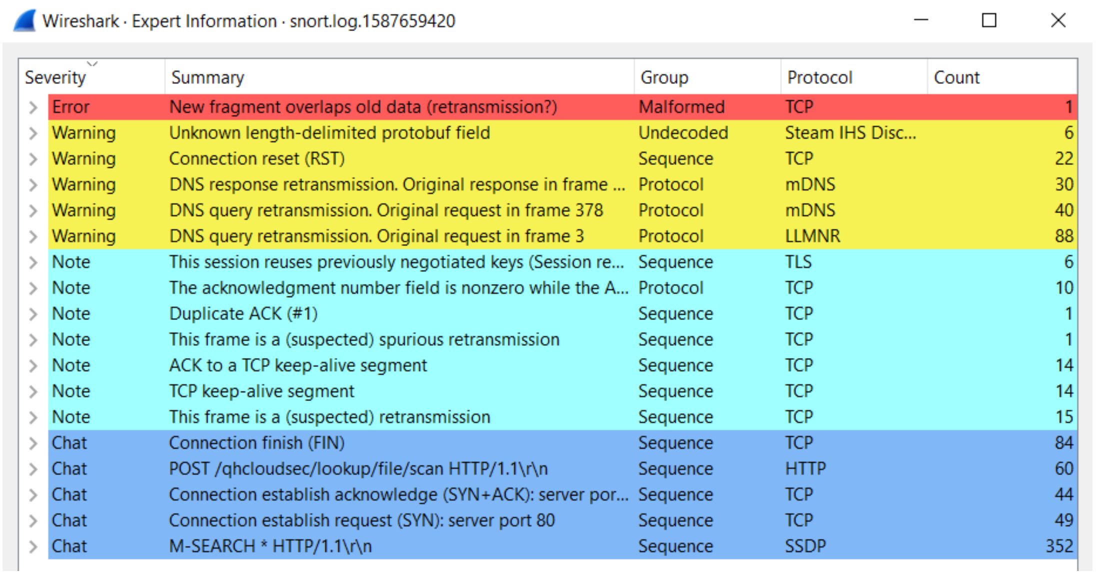
A line pattern of the packets is shown on the I/O graph, as seen in Figure 6. The x-axis is the time interval per second, and the y-axis is the packets captured in every 10 seconds.



Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field
☑	HTTP	http	🟩	Line	Packets	
☑	UDP	udp	🟦	Line	Packets	
☑	SSDP	ssdp	🟪	Line	Packets	

Figure 6. I/O graph of captured data packets

The purpose of expert information is to give you a clearer understanding of unusual and remarkable network activity and to render network problems for novices and experts faster than the manual analysis of the packet list [9].



Severity	Summary	Group	Protocol	Count
Error	New fragment overlaps old data (retransmission?)	Malformed	TCP	1
Warning	Unknown length-delimited protobuf field	Undecoded	Steam IHS Disc...	6
Warning	Connection reset (RST)	Sequence	TCP	22
Warning	DNS response retransmission. Original response in frame ...	Protocol	mDNS	30
Warning	DNS query retransmission. Original request in frame 378	Protocol	mDNS	40
Warning	DNS query retransmission. Original request in frame 3	Protocol	LLMNR	88
Note	This session reuses previously negotiated keys (Session re...	Sequence	TLS	6
Note	The acknowledgment number field is nonzero while the A...	Protocol	TCP	10
Note	Duplicate ACK (#1)	Sequence	TCP	1
Note	This frame is a (suspected) spurious retransmission	Sequence	TCP	1
Note	ACK to a TCP keep-alive segment	Sequence	TCP	14
Note	TCP keep-alive segment	Sequence	TCP	14
Note	This frame is a (suspected) retransmission	Sequence	TCP	15
Chat	Connection finish (FIN)	Sequence	TCP	84
Chat	POST /qhcloudsec/lookup/file/scan HTTP/1.1\r\n	Sequence	HTTP	60
Chat	Connection establish acknowledge (SYN+ACK): server por...	Sequence	TCP	44
Chat	Connection establish request (SYN): server port 80	Sequence	TCP	49
Chat	M-SEARCH * HTTP/1.1\r\n	Sequence	SSDP	352

Figure 7. Expert information on captured data packets

8. Conclusion

The primary problem that every organization faces in this digital era is network security. Network monitoring has now been an essential information protection aspect of defending attacks. IDS plays a vital role in network monitoring tools, such as Wireshark and Snort. Wireshark and Snort can monitor intrusive electronic information graphically on network processes or movements. IDS and IPS network management improves network connectivity efficiency and security.

In this paper, we have discussed the NIDS properties of Snort used for intrusion detection with the help of predefined rules and alert the user by sending alert messages. A log file is created by Snort that stores all the captured packets with their alert messages. The log file is exported to

Wireshark to analyze the captured network packets. Wireshark generates every single detail about the packets of the log file. It gives details about frames, Ethernet, internet protocols, protocol hierarchy, etc.An I/O graph summarized the flow of the packet. It displayed the total traffic in the capture file, usually measured in bytes or packets per second. Expert information gives details of all the protocols used during network analysis.

References

[1]. Rashmi Hebbar, Mohan K., "Packet analysis with Network Intrusion Detection System" International Journal of Science and Research (IJSR), vol. 4, Issue 2, February (2015).
[2]. V. Ndatinya, Z. Xiao, "Network Forensic Analysis using Wireshark", International Journal of Sensor Networks, Vol. 10, No. 2, (2015).
[3]. M. Ahmad, K. Ullah, "Snort Implementation with Wireshark HelpingAvoiding DDOS Attack in the Cloud Computing", International Journal of Computer Science and Information Security, Vol. 15, No. 1, January (2017).
[4]. Qing-Xiu Wu, "The Network Protocol Analysis Technique in Snort", International Conference on Solid State Devices and Materials Science, (2012).
[5]. Saboor, M. Akhlaq, B. Aslam, "Experimental Evaluation of Snort against DDOS Attacks under different Hardware Configurations", 2nd National Conference of Information Assurance, (2013).
[6]. Sakshi Singh, Suresh Kumar, "Capabilities of Wireshark as intrusion Detection System", International Journal of Recent Technology and Engineering, vol. 8, Issue 5, January (2020).
[7]. Sakshi Singh, Suresh Kumar, "Qualitative Assessment of Digital Forensic Tools" Vidyabharati International Interdisciplinary Research Journal, vol. 10, Issue 1, March (2020).
[8]. Snort_manual.pdf
[9]. https://www.wireshark.org/docs/wsug_html_chunked/ChAdvExpert.html

Authors

Sakshi Singh received her B.Tech degree in computer science and engineering from Guru Gobind Singh Indraprastha University (GGSIPU), Delhi, India, in 2016 and M.Tech (P) degree in information security from Ambedkar Institute of Advanced Communication Technology and Research (AIACTR), Delhi, India. Her research interests include digital forensics and information security.

Dr. Suresh Kumar received his PhD degree in computer science and engineering from Maharshi Dayanand University, Haryana, India and M. Tech and M. Sc degree in computer science from Kurukshetra University, Haryana, India. His major field of research is Semantic Web. His areas of specialization are semantic web, operating system, Information retrieval, and Database Management system. He is currently working as an associate professor with department of computer science and engineering, Ambedkar Institute of Advanced Communication Technology and Research (AIACTR), Delhi, India. He is the author, co- author of more than 45 publications in international/national journals and conferences.