NETWORK FORENSICS ANALYSIS

Report submitted for the partial fulfillment of the requirements for the degree of
Bachelor of Technology in
Information Technology

Submitted by

ARIJIT BISWAS

REGISTRATION NO –141170110114

UNIVERSITY ROLL NO -11700214019

BARUN KUMAR SINGH

REGISTRATION NO –141170110122

UNIVERSITY ROLL NO -11700214027

SUMIT CHOWDHURY

REGISTRATION NO –141170110172

UNIVERSITY ROLL NO -11700214077

Under the Guidance of
SUDARSAN BISWAS

(ASSISTANT PROFESSOR, RCC INSTITUTE OF INFORMATION TECHNOLOGY)

श्रमम् बिना न किमपि साध्यम्

RCC Institute of Information Technology
Canal South Road, Beliaghata, Kolkata – 700 015
[Affiliated to West Bengal University of Technology]

Acknowledgement

We would like to express our sincere gratitude to Mr. Sudarsan Biswas of the department of Information Technology, whose role as project guide was invaluable for the project. We are extremely thankful for the keen interest he / she took in advising us, for the books and reference materials provided for the moral support extended to us.

Last but not the least we convey our gratitude to all the teachers for providing us the technical skill that will always remain as our asset and to all non-teaching staff for the gracious hospitality they offered us.

Place: RCCIIT, Kolkata

Date:

ARIJIT BISWAS

REGISTRATION NO –141170110114

UNIVERSITY ROLL NO -11700214019

BARUN KUMAR SINGH

REGISTRATION NO –141170110122

UNIVERSITY ROLL NO -11700214027

SUMIT CHOWDHURY

REGISTRATION NO –141170110172

UNIVERSITY ROLL NO -11700214077

Department of Information Technology
RCCIIT, Beliaghata,
Kolkata – 700 015,
West Bengal, India

Approval

This is to certify that the project report entitled “**Network Forensic Analysis**” prepared under my supervision by, be accepted in partial fulfillment for the degree of Bachelor of Technology in Information Technology.

It is to be understood that by this approval, the undersigned does not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn thereof, but approves the report only for the purpose for which it has been submitted.

…………………………………………
Mr. Sudarsan biswas
Assistant Professor,
Dept. of Information Technology,
RCCIIT,Kolkata

Counter Signed by:

………………………………………..
Dr. Abhijit Das
Associate Professor & Head
Dept of Information Technology
RCC Institute of Information Technology,
Kolkata – 700015,
India

RCC INSITUTE OF INFORMATION TECHNOLOGY

KOLKATA– 7OOO15, INDIA

श्रमम् बिना न किमपि साध्यम्

CERTIFICATE of ACCEPTANCE

The report of the Project titled Rule Based Network Forensic Analysis method submitted Arijit Biswas(11700214019), Barun Kumar Singh (11700214027),Sumit Chowdhury (11700214077) of B.Tech.(IT) 8th Semester of 2018 has been prepared under our supervision for the partial fulfillment of the requirements for B Tech (IT) degree in Maulana Abul Kalam Azad University of Technology.

Name of the Examiner **Signature with Date**

1. ……………………………………. …………………………………………

2. ……………………………………. ………………………………………….

3. ……………………………………. …………………………………………..

4. ……………………………………. ……………………………………………

5.……………………………………… …………………………………………..

6.……………………………………… ……………………………………………

7.……………………………………… ……………………………………………

1. ABSTRACT

Rule based network forensic analysis is a process which analyzes intrusion evidence captured from net-worked environment to identify suspicious entities and stepwise actions in an attack scenario. Flooding attack is one of the serious threats of network security on Web servers that resulted in the loss of bandwidth and overload for the user and the service provider web server. The first step to recognizing the network flooding attack is by applying the detection system Intrusion Detection System (IDS) like Snort and Wireshark. By capturing the traffic in net-worked environment helped in generating the I/O graph and flow graph and by generating rules we declare some alert from different protocols like TCP/IP and ICMP. After generating the alert file, we generate an attack graph. Security analysts use attack graphs for detection, defense and forensics. And from the attack graphs, we will analyze the port id as nodes. If any port flooding the network. Then it will declare as an attacker machine or a malicious packet/port.

2. INTRODUCTION

The current improvements in modern technology have enabled the use of computer systems in conducting business and in gathering and sharing information in corporations and academic institutions using the Internet [1]. Today, banks make use of networks to perform its financial operations, hospitals have the records of their patients in databases, and many companies has been presented on the Internet, so that any user with Internet access is able to choose the product that he/she desires and buy it online. The data that is handled in this type of businesses should be saved from attacks.

The Transmission Control Protocol and Internet protocol (TCP/IP), which is the protocol that Internet and many of today's networks based on, was first developed in 1979. The primary focus was to ensure reliable communications between groups of networks connected by computers acting as gateways. At that time, security was not a primary concern due to the size of this Internet and that most of the users knew each other. However, the base technologies used to construct this network contained many insecurities, most of which still exist today. Due to a number of well reported attacks on private networks originating from the Internet, security has become a primary concern for organizations connecting to the Internet. Organizations need to securely conduct business and protect their data and computing from attacks. Such needs are heightened as businesses link geographically distant parts of the organization using private networks based on TCP/IP. Nowadays, guarantee of secure communication is as important as the traditional computer and information security assurance. Information in transit (as messages) must be protected from unauthorized release and modification, and the connection itself must be established and maintained securely. Prevention of illegitimate traffic is one of the goals of communication security and seeks to prevent an eavesdropper from gaining any meaningful information about network users' behavior or objectives by observing the legitimate traffic on the network.

To protect the enterprise, security managers have deployed a variety of technologies. While these technologies are useful for defending corporate assets, they have limitations. For example, firewalls may be configured to block certain types of traffic, but attackers still find ways to exploit legitimate traffic types to mount their attacks.

2.1 Problem Definition

Network forensics analysis is the capture, recording, and analysis of network events in order to discover the source of security attacks or other problem incidents. We have to capture the malicious packets mainly in data link layer, among thousands of packets with the help of certain tools/software's like Wireshark, Snort by Applying some rule & signature-based algorithms in Intrusion Detection System. And, we have to concentrate mainly in three types of attacks i.e. DoS, DDoS, MAC-Flooding Attack and Man-in-the-middle attacks.

2.2 Objective

The main objective of this project work is to design and develop of a rule-based Network Intrusion Detection System which can detect intrusions based on certain rules and can also detect novel attacks which are Anomalous in nature. The work also aimed at reducing number of false alarms by characterizing the target network with appropriate network parameters and analyzing them. So in this work, rule based detection technique is used for discriminating the anomalous attacks from that of normal activities The project is integrated with a open source signature based IDS called SNORT so that it forms a complete package having both signature and anomaly techniques for effective defense against the Network attacks. To implement network intrusion detection system based on rules or signature; we need to install some tools along with snort such as Win Cap, Wireshark etc. snort is installed in the computer within the network. Once it's installed completely it will automatically capture the network packet which are passed over the network. Identification of attack in snort based on protocols and that protocols categorized into four groups (TCP, UDP, IP and ICMP protocol). We proposed a rule-based Network IDS which will examines ongoing traffic, transactions, activity, or behavior for matches with known patterns of events specific to known attacks. Rule- based detection system (also called misuse based), very effective against known attack, it implies that misuse detection requires specific knowledge of given intrusive behavior. The advantages of rule-based network Intrusion detection system is, it produces low false positives, and it is easy to use.

2.3 Motivation

This era is completely depending on computer and network in any form (like social media, E-marketing, E-banking etc.), and today's in the field of network security, Intrusion Detection System (IDS) playing an important role to secure network infrastructure. Whenever we are talking about security, network security is the big challenge among the researchers and most researchers are working in the field of network security.

The purpose of network security is to protect the network from unauthorized access and disclosure, but till now we did not get the perfect solution for network security. In network security area there are different tools (as a software and hardware) are available such as antivirus, firewall etc. but they are not able to cover all security risk.
The main work of intrusion detection system is to collect the packet from network, process it and if attack identifying then It will generate an alert for possible attack.
Network security, intrusion detection system has two flavours for both Network and Host based categories and that's flavours known as signature or rule bases intrusion detection and anomaly based intrusion detection. Signature based intrusion detection system also known as misuse detection, and the essence of misuse detection centres around using an expert system to identify intrusions based on a predetermined knowledge base.
Our main motivation to take up the project is to learn, test and analyse various network-based attacks using the rule-based detection techniques with the help of latest rules (till date) available in the snort community so that we can generate alert for some latest invented attacks too.

3.Literature Survey

3.1 Network Forensics Analysis Background

Network forensics is the sub-branch of digital forensics which deal with analysis of computer network traffic for the purposes of information gathering, legal evidence, or intrusion detection. It involves the process of capturing, storing, and analysis of network events. It is sometimes also called packet mining, packet forensics. Regardless of the name, the idea is the same: record every packet of network traffic (all emails, all database queries, all Web browsing–absolutely all traffic of all kinds traversing an organization's network) to a single searchable repository so the traffic can be examined in detail [2].

Network forensics is a comparatively new field of forensic science. The growing popularity of the Internet in homes means that computing has become network-centric and data is now available outside of disk-based digital evidence. Network forensics can be performed as a standalone investigation or alongside a computer forensics analysis (where it is often used to reveal links between digital devices or reconstruct how a crime was committed).

Marcus Ranum is credited with defining Network forensics as "the capture, recording, and analysis of network events in order to discover the source of security attacks or other problem incidents" [3].

Compared to computer forensics, where evidence is usually preserved on disk, network data is more volatile and unpredictable. Investigators often only have material to examine if packet filters, firewalls, and intrusion detection systems were set up to anticipate breaches of security.

Systems used to collect network data for forensics use usually come in two forms [4]:

- "Catch-it-as-you-can" – This is where all packets passing through a certain traffic point are captured and written to storage with analysis being done subsequently in batch mode. This approach requires large amounts of storage.

- "Stop, look and listen" – This is where each packet is analyzed in a rudimentary way in memory and only certain information saved for future analysis. This approach requires a faster processor to keep up with incoming traffic.

The basic architecture of the network forensics analysis process is shown below:

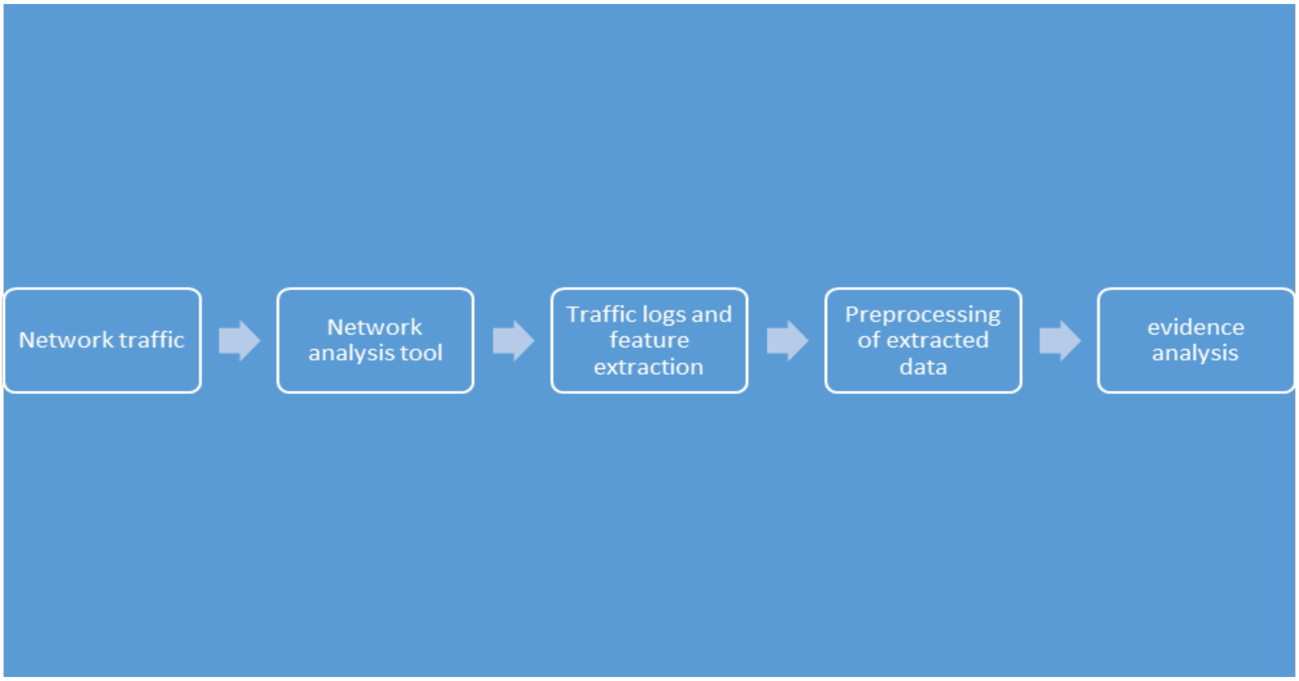

Network traffic
Network analysis tool
Traffic logs and feature extraction
Preprocessing of extracted data
evidence analysis

Fig 1. Basic architecture of the network forensics analysis process

During Initial Phase, analysis of the network traffics will be done using network analysis tools. The network analysis tools will perform various analysis such as packet analysis, web request analysis etc. After completion of analysis, a report will be generated containing all kinds of traffic logs and various features of the network-based APIs calls which can be suspicious. The extracted data are preprocessed as not all data present in the report will be suspicious data. In the final stage, analysis of the evidence collected in the form of suspicious data is performed.

To perform network forensics analysis, the tools employed must provide three essential capabilities: capturing and recording data, discovering data, and analyzing data.

• **Capturing and Recording Data:** This is the ability to capture and store multiple terabytes of data from high throughput networks (including 10G and even 40G networks) without dropping or missing any packets. Every network forensic solution has its limitations, including sustainable throughput, packets per second, data management, search functions, etc. These limitations can and should be determined through practical lab tests, and the results should be repeatable and documented.

• **Discovering Data:** Once data are recorded on the storage media, the solution should provide a means of filtering particular items of interest, for example, by IP address, application, context, etc. IT engineers rely on discovery tools for sifting through terabytes of data to find specific network conversations or individual packets in a timely fashion.

Analyzing Data: To further accelerate discovery and analysis, IT engineers benefit from a forensics solution's built-in assistance for examining the patterns and anomalies found during the discovery process. Automated analysis, including Expert analysis that explains the context of network events, helps IT engineers quickly identity anomalous or otherwise significant network events.

There are various tools employed for network traffic monitoring and analysis such as:

- TCPDUMP
- WINDUMP
- SNORT
- TCP WRAPPER
- NETFLOW
- WIRESHARK

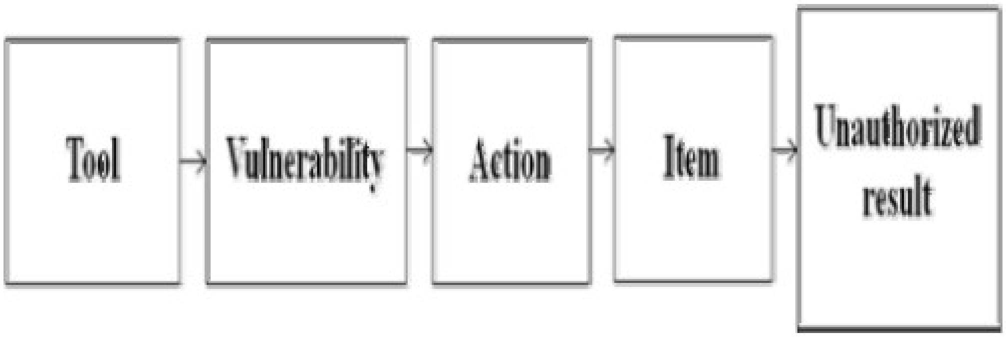
However, we are mainly focused on using a tool called SNORT, an open source tool for Network Intrusion Detection System.

3.2 TYPES OF NETWORK ATTACKS

3.2.1 Attack

Due to the great development of computer technology to generate informatics crimes, many people are trying to harm the computer system or network, where the goal is to attack the computer systems to obtain the desired results such as theft of personal identities, disable online business, generating traffic in a network unexpectedly, delete or extract confidential information, obtain identification of access source, generate viruses or worm without authorization.

An attack is a series of steps from attacker to achieve an unauthorized result [5]. An attack generally is composed of five parts, which form part of a logical algorithm of an attacker as shown below:



In figure shown the elements carried out by an attacker. An attacker uses a tool to exploit a vulnerability to perform an action on a target in order to achieve an unauthorized result. To be successful, an attacker must find paths that can be connected (attacks), perhaps simultaneously or repeatedly.

Network security is sometimes more than what people always thought it to be, malware, virus, Trojan, hackers. Network security could be caused by unintentional human error and it could be compromised by human nature as well. With increasing reliance on computer systems worldwide for data processing and information storage, the need for legitimate security of information and data cannot be overemphasized. Un-authorized access, revelation or destruction of data can violate individual privacy and even threaten the existence of an organization. Since information is regarded as the live wire of an organization, it is, therefore, necessary to secure computer systems and the stored information.

Quick information accessibility on the Internet has become increasingly important for growing businesses. As companies begin to spread various business functions to the public network, precautions are highly needed to make sure that their network not been tampered with or does not fall to wrong hands. If a network is accessed by a hacker or dissatisfied employee, it could create havoc for organization proprietary data, affect company productivity negatively, and retard the ability to compete with other businesses. Unauthorized network access can also harm a company’s relationship with customers and business partners who may question the company's ability to protect their confidential information. Furthermore, any part of a network can be susceptible to attacks or unauthorized activity as earlier discussed. Company competitors or even internal employees can violate all routers switches and hosts. In order to determine the appropriate ways of protecting a company's property against attackers, the Information Technology Manager of such company should understand the attacks that can be instigated and the havoc they can cause to business infrastructures.

3.2.2 CATEGORIES OF SECURITY THREATS

Security threat can be categorized into four parts and these categories are the ways or forms through which threats can be carried out on a network.

I.UNSTRUCTURED THREATS

Unstructured security threat is the kind of threat created by an inexperienced person trying to gain access to a network. They commonly use common hacking tools, like shell scripts, and password Crackers. A good security solution should easily thwart this kind of attack. In other words, these kinds of hackers could not be underestimated because they can cause serious damage to network.

II. STRUCTURED THREATS

Unlike unstructured threats, structured threat hackers are well experienced and highly sophisticated. They use sophisticated hacking tools to penetrate networks and they can break into government or business computers to extract information. On certain occasions, structured threats are carried out by organized criminal gangs or industry competitors.

III. EXERNAL THREATS

Some unauthorized people outside the company who do not have access to the company’s computer system or network could cause external threat. They usually break into company's network via the Internet or server. Both experienced and inexperienced hackers could pose external threats.

IV.INTERNAL THREATS

This kind of threat could be by a disgruntled employee who has authorized access to the company's network. Like external threats, the damage that could be caused by such a hacker depends on the expertise of the hacker.

Technically competent hackers have been able to fashion a structured attack targeted at communication protocols. The OSI model has seven layers that are used for communication between networking devices, which are with vulnerabilities that can be controlled. Basically, higher layers cannot be secured while the lower layers are also not being secured, yet in recent years there has been limited attention to insecurities at the physical layer or data link layer despite changes in network operational practice that include developments like nation-wide layer two networks and national and regional optical networks [6].

Currently known threats at lower levels of the OSI stack include ARP spoofing, MITM (man-in-the-middle) attacks at layer two, and physical layer attacks such as passive optical taps or the interception of wireless network signals by attackers. While these attacks are well known, little research is currently focused on addressing those concerns.

1. PHYSICAL LAYER

The physical layer is responsible for transferring data over network communication media. It could also be referred to as most changeable and vulnerable layer. When dealing with this type of layer, unserious incidents like unplugging the computer power cord or removing a network cable could sometimes cause a great and untraceable havoc on a particular network, and it could cause great damage to the computer.

There are plenty of vulnerabilities that the physical layer is facing, few of which include: loss of environmental control, damage of hardware and data, disconnection of physical data links, power loss, input logging like keystroke and other physical theft of data and hardware, and undetectable interception of data. These vulnerabilities could cause great damage to network security through physical layers if prevention is not done at the right time. Nevertheless, there are always solutions available for any threat of damage caused to a network.

As mentioned above, there are always solutions for every problem. Perimeters and enclosures lock, electronic lock mechanisms for logging and detailed authorization, data storage cryptography, PIN and password secured locks, electromagnetic shielding, biometric authentication systems, and video and audio surveillance can all be used to prevent or secure any threat that is coming to attack a network or that has attacked a network via the physical layer.

2. DATA LINK LAYER

This is the layer where transmission of data packets has been prepared by the physical layer. Communication of the data link is somehow weak in terms of security. The key component at layer 2 communications is the switch, which is also used for communication at layer 3. Data link is susceptible to many layer 3 attacks. The prime example of the layer 2 element is 'wardriving' the method of going around searching for wireless LAN (802.11) Network with default security settings. VLAN in layer 2 switches are also vulnerable to attacks.

All the OSI layers face different threat that affect them at their various stages. Highlighted below are the problems faced by layer two of the OSI model and the solution to the problems.CAM (Content-Addressable Memory) table overflow, MAC (Media access control) spoofing, STP (Spanning Tree Protocol) Manipulation, ARP (Address Resolution Protocol) attack, and VLAN hopping are the problems faced by data link layers. CAM can be controlled by configuring port security on switch in order to provide a MAC address specification on a particular switch port so that it can be learnt and memorized by the port to detect an invalid address on the port.

Like in CAM, port security commands can also be used to control MAC-spoofing attacks. The command can allow the switch to specify a protection action whenever violations of port-security occur.

ARP attacks can be mitigated by using Hold-down timers in the configuration interface menu. This can be achieved by setting an entry-stayed time in the ARP cache.

Control of VLAN hopping could be done by issuing VLAN IDs for trunk ports, and disabling of unused switch port and putting them in an unused VLAN.

3. NETWORK LAYER

The network layer is a medium used by packets to get to their final destination over multiple data. As said earlier in the previous chapter above, virtually all the layers have challenges of security. The lowest third layer of the OSI model is known to face challenges of information privacy problems and Denial of Service attacks. Internet protocol (IP) is the well-known protocol for the network layer. There are many security risks associated with the IP in the network layer.

The part of the security risk affecting network layers are network layer packet sniffing, route spoofing, IP Address spoofing.

Route policy controls - This mitigation gives a network administrator total control over the routing behaviour of particular system. This control also improves network stability.

Authentication— Packet sniffing can be mitigated by various methods, and the using of strong one-time passwords is one mitigating method It could also be controlled by deploying switch infrastructure to counter the use of packet sniffers.

4. TRANSPORT LAYER

The transport layer makes use of mechanisms such as TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) to provide end-to-end communication services, which allow data to completely arrive at its destination. Poor handling of undefined conditions is one the problems this layer is facing. Overuse of a particular port for multiple functions could also be a vulnerability of transport layers as well as poor handling of undefined conditions, transport protocol implementation differences, transport-layer mechanisms overloading.

Firewall rules that can be used to limit access to specific transmission protocols and sub protocol information should be strict.

Other measures include preventing out-of-state packets, by inspecting the layer at firewall from entering the perimeter and preventing the attacker and takeover of communications by implementing stronger transmission and layer session identification mechanisms.

5. SESSION LAYER

The session layer keeps track of data communications and organizes them into a logical flow. This layer also establishes, manages, and terminates sessions between applications and manages the data exchange between presentation layer entities. Attackers can cause damage to company`s network through this medium by unlimited attempts to guess the password, and they can as well make use of cruder methods to exhaust possible password strings. Weakness of used authentication mechanisms, hijacking and spoofing of session identification, failed authentication attempts could lead to information leakages, and unlimited failed sessions can help attackers to accessing credentials.

The following precautions should be put in place so as to prevent the error from happening or to eradicate it if happed already. Passwords should be well encrypted and change on a regular basis, there should be a specific expiry data for a particular user account for regular monitoring, session identification information should be protected through cryptographic means, the use of timing machine is encouraged for limiting failed session attempts.

6. PRESENTATION LAYER

The presentation layer deals with service request responsibility from the application layer and service request issuing to the session layer. The presentation layer is known for three functions: encoding and decoding data, encrypting and decrypting data, compressing and decompressing data. Although presentation layer is one of the most secured layer among the OSI model, it has its own threats. The threats common to this layer are fake certificate attacks and man-in-the-middle attacks. Care should be taken when handling unexpected input, because it can crash applications, Privacy protection could be exploited by cryptography flaws and remote manipulation or information leakage could occur when using external supply input unintentionally. The solution that should be put in place to counter the above-mentioned vulnerabilities include input coming into the application function should be carefully specified and checked; separating user input and program control functions; cryptography solutions should be reviewed continuously to ensure current security versus emerging threats.

7. APPLICATION LAYER

The application layer is the closest to the end user and it allow users to interact with the application and the networks. This interface could be a prime target for unauthorized use and abuse over the network if the application is weak or unauthenticated. For instance, an intruder has no challenge in guessing file names in TFTP protocol, because username or even password is not required to access files in the TFTP protocol.

Standard security control is bypassed through the backdoors and application design. If security controls force approach is not adequate, it results in excessive access or insufficient access; when application security is too complex, it is sometimes difficult for users to understand; and program logic flaws could sometimes cause programs to crash or undesired behavior.

The use of application level access controls in order to define access to application resources, use of baseline in measuring application implementation; such as application codes reviews and standard testing. Using of host-based firewall systems to regulate traffic, application activities and inquiries monitored by the use of IDS systems are all means to control the vulnerabilities of application layers.

3.2.3 TYPES OF ATTACKS

RECONNAISSANCE ATTACKS

Administrators could overlook this attack because of the form it takes to penetrate the network. It always makes this kind of noise that might let the administrator to think is just a network noise. A reconnaissance attack is always used by hackers to gather information about a particular targeted network, which they subsequently used to access the network or as DoS attacks.

1) PACKET SNIFFERS

As its name implies, a packet sniffer is a very good device used by the administrators for detecting any kind of fault in the network. As it is a good device for administrators for monitoring or analyzing a network, so is it a good device for attackers for capturing packets sent across networks.

2) PORT SCANS AND PING SWEEP

These applications run a series of tests against hosts and devices to identify vulnerable services that need to be attended to.

These attacks can attempt to:

I. Identify all services on the network.
II. Identify all hosts and devices on the network.
III. Identify the operating systems on the network.
IV. Identify vulnerabilities on the network.

3) INTERNET INFORMATION QUERY

“WHOIS” is the Internet weapon attackers use to view addresses by DNS queries so that they can present a targeted company live host. By querying the IP addresses, some information could be revealed, such as the range of addresses and domain associated to those addresses. All revealed information could prompt an attacker to carry out whatever attack they intend to do.

ACCESS ATTACK

Access attackers could be outsider hackers or inside users gaining entrance into a network in an unauthorized way to steal some vital and confidential information from the systems. They could also engage in destruction of resources so that some information that could lead to them could not be seen. There are different reasons for different attacks. Intruders use access attacks on networks or systems for the following reasons: to retrieve data, to gain access and escalate their access privileges. Access attacks can consist of the following:

1) PASSWORD ATTACKS

Hashes of passwords could be taken by L0phtCrack and the clear-text passwords could be generated from them; a brute-force password attack offers access to accounts that can be used to alter critical network services and files. A typical example for such attack that compromises the network integrity is when an attacker modifies the network’s routing tables. By doing so, the attacker ensures that all network packets are routed to the attacker before being transmitted to their final destination. In such cases, an intruder can monitor all network traffic. There are two methods for computing passwords with L0phtCrack:

I. Dictionary cracking: The password hashes for all words in a dictionary file are compared and computed against all of the password hashes for the users. This is an extremely fast method that finds very simple passwords.

II. Brute-force computation: In this method, particular character sets are used, such as A to Z plus 0 to 9 or A to Z and compute the hash for every potential password made up of those characters. Brute-force compilation usually computes passwords if those passwords are made up of the character set someone has selected to test. The problem for the attacker is the time required for the completion of this type of attack.

2) TRUST EXPLOITATION

Trust exploitation is a situation where by an individual is taking advantage of a trustable and reliable relationship within a network. An example of such an attack is a perimeter network connected to a corporate network. Hacker leverages on the existing trust relationships. Several trust models that exist:

I. Windows

II. NIS+

III. Active directory

IV. NIS

V. Linux and UNIX

3) PORT REDIRECTION

Port redirection attacks are a type of trust exploitation attack, which uses a host that is fragile in passing traffic that would otherwise be dropped via a firewall. A host on the outside can contact the host on the public services segment (mostly known as the demilitarized zone [DMZ]) (Host A), but not the host on the inside (Host B). The host on the public services segment can be reached

by the host on both the inside and outside. If hackers successfully compromise the public services segment host, they will be able to install software to channel traffic from the outside host directly to the inside host. Even though neither communication fails to agree with the rules implemented in the firewall, the outside host has now achieved a good network connectivity to the inside host simply through the port redirection process on the public services host. A good example of an application that can render this kind of access is Netcat.

3.3 Attacks on OSI Layer 2 (Data Link Layer)

Our work related to network forensics mainly focuses on the detection and the analysis of network-based attacks based on Data link layer of the OSI layer.

There are mainly three attacks which are based on Data link layer. Other possible attacks are the cause of these three attacks.

1. MAN-IN-THE MIDDLE ATTACK [7]

A man-in-the-middle attack necessitates that the hacker possess access to network packets that come via a network. A man-in-the-middle attack could be implemented using network packet sniffers and routing and transport protocols.

Theft of information, hijacking of an ongoing session to gain access to internal network resources, traffic analysis to derive information about the network and its users, denial of service, corruption of transmitted data, and introduction of new information into network sessions are possible tools uses by man-in-the-middle attacks to attack a network.

Someone working for an ISP can gain access to all network packets and perform all of the above operations.

The below example shows how Attacker Peter is modifying the keys of both the victims. Due to modification in the keys, the attacker is able to perform the Man-in-the-Middle Attack.

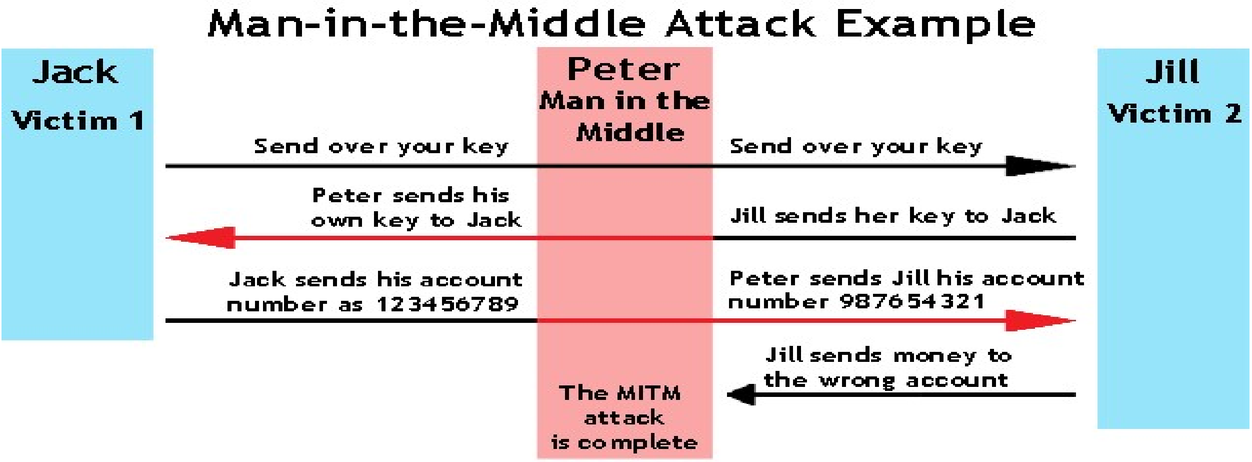

Man-in-the-Middle Attack Example
Jack
Victim 1
Peter
Man in the Middle
Jill
Victim 2
Send over your key
Send over your key
Peter sends his own key to Jack
Jill sends her key to Jack
Jack sends his account number as 123456789
Peter sends Jill his account number 987654321
Jill sends money to the wrong account
The MITM attack is complete

Fig 3. Man in the Middle Attack

2. DENIAL OF SEVICE ATTACKS [8]

A denial-of-service attack is a security event that occurs when an attacker takes action that prevents legitimate users from accessing targeted computer systems, devices or other network resources[1]. It is an attack meant to shut down a machine or network, making it inaccessible to its intended users. DoS attacks accomplish this by flooding the target with traffic or sending it information that triggers a crash. In both instances, the DoS attack deprives legitimate users (i.e. employees, members, or account holders) of the service or resource they expected. [2]

In a denial-of-service (DoS) attack, an attacker attempts to prevent legitimate users from accessing information or services. By targeting your computer and its network connection, or the computers and network of the sites you are trying to use, an attacker may be able to prevent you from accessing email, websites, online accounts (banking, etc.), or other services that rely on the affected computer. The most common and obvious type of DoS attack occurs when an attacker "floods" a network with information. When you type a URL for a particular website into your browser, you are sending a request to that site's computer server to view the page. The server can only process a certain number of requests at once, so if an attacker overloads the server with requests, it can't process your request. This is a "denial of service" because you can't access that site. [3]

Denial-of-service attacks are noisy by design, and they intend to make a statement. They're not subtle attempts to infiltrate a Web site's defenses, which can be much more insidious because that gives hackers access to whatever confidential information is stored there. [5]

The function of a denial of service attack is fundamentally to flood its target machine with so much traffic that it prevents it from being accessible to any other requests or providing services. The target machine is kept so busy responding to the traffic it is receiving from its attacker that it has insufficient resources to respond to legitimate traffic on the network.

A distributed denial of service (D-DOS) attack adds a many-to-one dimension to these forms of attacks. This form of denial of service generally involves a machine containing a master program and several machines which have been enslaved as zombie machines. They are referred to as zombies as these machines which are originally the victim of a denial of service attack unwittingly become an attacker. These zombies or daemons reside on the victim's machine until they are instructed by the master machine to attack another target. This makes it almost impossible to track down the real attacker as the attack is coming from zombie machines which have no knowledge of the origin of the attack.

➢ How do you know if an attack is happening?

Not all disruptions to service are the result of a denial-of-service attack. There may be technical problems with a particular network, or system administrators may be performing maintenance. However, the following symptoms *could* indicate a DoS or DDoS attack:

- unusually slow network performance (opening files or accessing websites)
- unavailability of a particular website
- inability to access any website
- dramatic increase in the amount of spam you receive in your account [3]

- **How a "denial of service" attack works**

In a typical connection, the user sends a message asking the server to authenticate it. The server returns the authentication approval to the user. The user acknowledges this approval and then is allowed onto the server. In a denial of service attack, the user sends several authentication requests to the server, filling it up. All requests have false return addresses, so the server can't find the user when it tries to send the authentication approval. The server waits, sometimes more than a minute, before closing the connection. When it does close the connection, the attacker sends a new batch of forged requests, and the process begins again--tying up the service indefinitely. [4]

- **How does an attacker launch a Denial of Service attack?**

There are many different ways that an attacker can launch a Denial of Service attack. They range from simply unplugging a server from the network (if they have physical access) to coordinating large armies of zombie computers to launch a large scale distributed attack against their target using:

- Buffer overflows in the application functions
- Malformed data to raise unexpected exceptions
- Exploited race conditions in multi-threaded systems
- Heavy-duty SQL queries via web forms and "spamming" them with requests, e.g., inserting % characters within search query fields
- SQL Injection attacks executing recursive CPU-intensive queries
- The end users' web browsers to overload the application with parallel requests via persistent / reflected Cross-Site Scripting attacks
- Overly-complex regular expressions within search queries
- Excessively large files uploaded to the server.[6]

- **Preventing Denial of Service Attacks**

With dotDefender web application firewall you can avoid DoS attacks because dotDefender inspects your HTTP traffic and checks their packets against rules such as to allow or deny protocols, ports, or IP addresses to stop web applications from being exploited.

Architected as plug & play software, dotDefender provides optimal out-of-the-box protection against DoS threats, cross-site scripting, SQL Injection attacks, path traversal and many other web attack techniques. The reasons dotDefender offers such a comprehensive solution to your web application security needs are:

- Easy installation on Apache and IIS servers
- Strong security against known and emerging hacking attacks
- Best-of-breed predefined security rules for instant protection
- Interface and API for managing multiple servers with ease
- Requires no additional hardware, and easily scales with your business.[6]

Methods of Denial of Service Attacks:

I. **Smurf attack** involves an attacker sending a large amount of Internet Control Message Protocol (ICMP) echo traffic to a set of Internet Protocol (IP) broadcast addresses. The ICMP echo packets are specified with a source address of the target victim (spoofed address). Most hosts on an IP network will accept ICMP echo requests and reply to them with an echo reply to the source address, in this case, the target victim. This multiplies the traffic by the number of responding hosts. On a broadcast network, there could potentially be hundreds of machines to reply to each ICMP packet. The process of using a network to elicit many responses to a single packet has been labeled as an "amplifier". There are two parties who are hurt by this type of attack: the intermediate broadcast devices (amplifiers) and the spoofed source address target (the victim). The victim is the target of a large amount of traffic that the amplifiers generate. This attack has the potential to overload an entire network

II. **SYN Flood** attack is also known as the Transmission Control Protocol (TCP) SYN attack and is based on exploiting the standard TCP three–way handshake. The TCP three-way handshake requires a three-packet exchange to be performed before a client can officially use the service. A server, upon receiving an initial SYN (synchronize/start) request from a client, sends back a SYN/ACK (synchronize/acknowledge) packet and waits for the client to send the final ACK (acknowledge). However, it is possible to send a barrage of initial SYN's without sending the corresponding ACK's, essentially leaving the server waiting for the non-existent ACK's. Considering that the server only has a limited buffer queue for new connections, SYN Flood results in the server being unable to process other incoming connections as the queue gets overloaded

III. **UDP Flood** attack is based on UDP echo and character generator services provided by most computers on a network. The attacker uses forged UDP packets to connect the echo service on one machine to the character generator (charges) service on another machine. The result is that the two services consume all available network bandwidth between the machines as they exchange characters between themselves. A variation of this attack called ICMP Flood, floods a machine with ICMP packets instead of UDP packets.

The figure shown below depicts how the attacker is broadcasting a ping request in the network backbone and rest of the network sends packet to the victim as a reply packet resulting in congestion of the victim's network.

- **Buffer overflow attacks** – the most common DoS attack. The concept is to send more traffic to a network address than the programmers have built the system to handle. It includes the attacks listed below, in addition to others that are designed to exploit bugs specific to certain applications or networks
- **ICMP flood** – leverages misconfigured network devices by sending spoofed packets that ping every computer on the targeted network, instead of just one specific machine. The network is then triggered to amplify the traffic. This attack is also known as the smurf attack or ping of death.

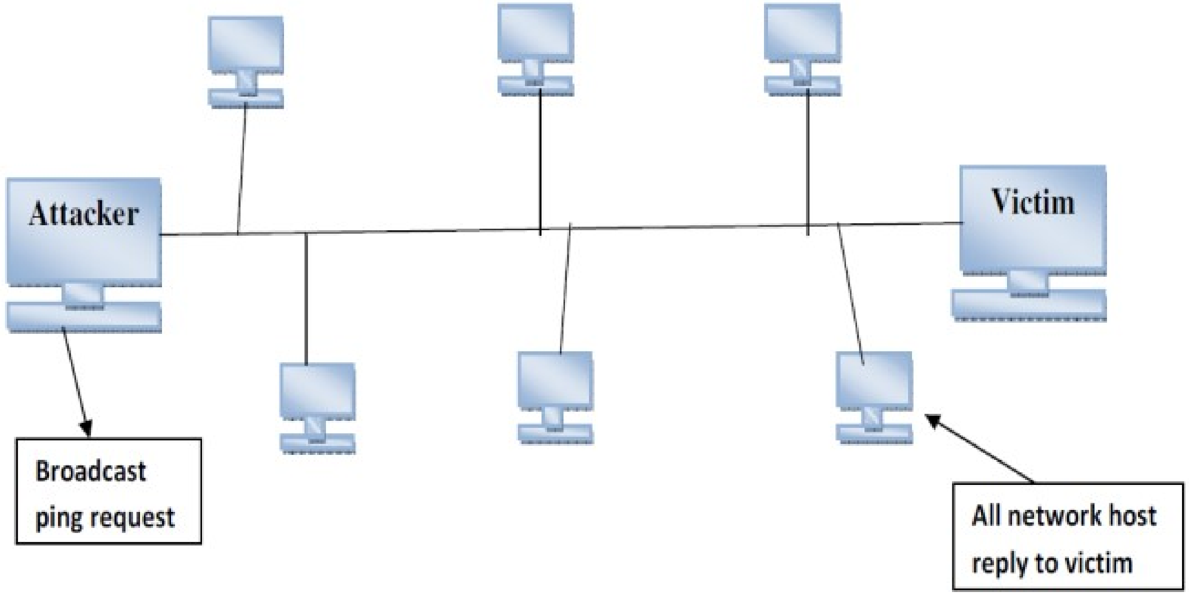
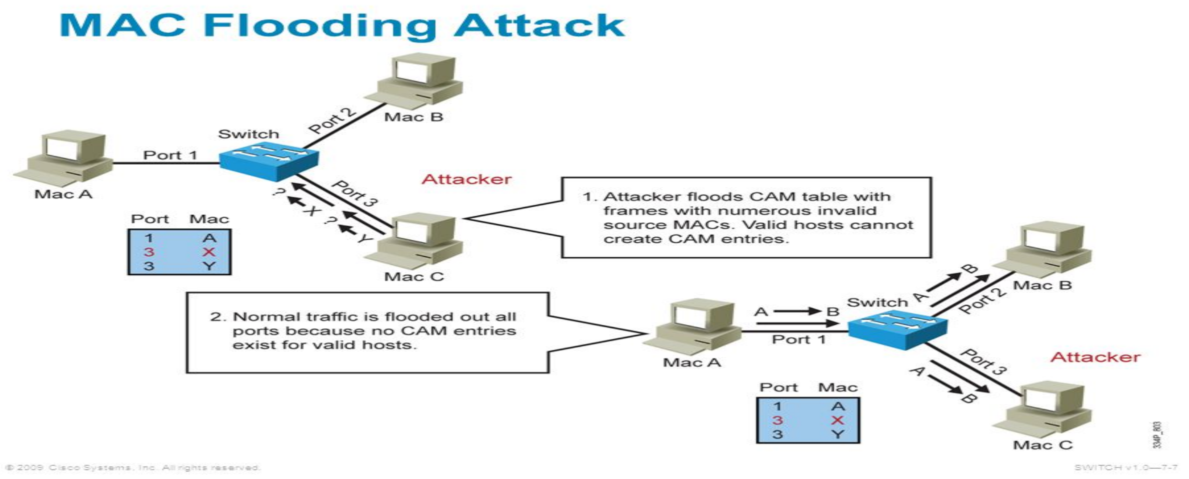


Fig 4. UDP Flood Attack

3. MAC FLOODING ATTACK [9]

In computer networking, MAC flooding is a technique employed to compromise the security of network switches. Essentially, MAC flooding inundates the network switch with data packets that disrupt the usual sender to recipient flow of data that is common with MAC addresses. Switches maintain a MAC (sometimes called as CAM) Table that maps individual MAC addresses on the network to the physical ports on the switch. This allows the switch to direct data out of the physical port where the recipient is located, as opposed to indiscriminately broadcasting the data out of all ports as a hub does. The advantage of this method is that data is bridged exclusively to the network segment containing the computer that the data is specifically destined for. In a typical MAC flooding attack, a switch is fed many Ethernet frames, each containing different source MAC addresses, by the attacker. The intention is to consume the limited memory set aside in the switch to store the MAC address table. This cause switches to enter into fail open state, in which switch will acts as hub. Attacker then can use packet sniffer to capture sensitive data. Some advance switch such as Cisco offers you protection against this attack.To prevent MAC flooding one of the following features should be configure in switch. Port security: Port security should be configured which limits number of MAC addresses that can be learned on ports connected to end stations. Implementations of IEEE 802.1X suites: It often allow packet filtering rules to be installed explicitly by an AAA server based on dynamically learned information about clients, including the MAC address.



MAC Flooding Attack

1. Attacker floods CAM table with frames with numerous invalid source MACs. Valid hosts cannot create CAM entries.

2. Normal traffic is flooded out all ports because no CAM entries exist for valid hosts.

© 2009 Cisco Systems, Inc. All rights reserved.

SWITCH v1.0—7-7

Fig 5. MAC flooding attack

3.4 MITIGATION OF NETWORK BASED ATTACKS

Information security managers have utilized multiple technologies to keep their networks safe. However, as an effect of the improvements in technology, networks are now connected to one or more outside networks – including, of course, the Internet. Hence, the corporations face with a wide range of threats. The fact that internal systems are actually quite vulnerable to all kinds of exploits makes these threats even worse. Plus, the widespread availability of reconnaissance tools has made it easier than ever for even novice attackers to bypass the enterprise security. So, security managers are under a lot of pressure to prevent any penetration to the network perimeter. Luckily, similar to the physical security, there are numerous security tools to help security managers in setting up complex protection strategy plans for their computer systems. Mostly common ones are commented subsequently.

Firewalls [10]

Firewalls are usually the first component of any perimeter defense. Firewalls provide a barrier of security among networks of different levels of confidence or security, utilizing network level access control politics. The major functional requirement of a firewall is to protect a private (internal) network from unauthorized external access.Firewalls act like traffic cops and perform the critical task of filtering traffic crossing the network boundary. This filtering is done according to predefined security policies, which can be specified at the network layer and/or at the application layer. Firewalls utilize these static, manually configured, security policies to differentiate legitimate traffic from non-legitimate traffic. Typical reasons for using a firewall to protect a private network include the following:

- To prevent unauthorized external users from accessing computing resources on the internal network. This is necessary because it is extremely difficult and costly to attempt to secure all the hosts within a private network

- To control internal user access to the external network to prevent the export of proprietary information,

- To provide a dependable and reliable connection to the Internet, so that employees do not implement their own insecure private connections.

Set of rules specifies which packets can pass, which cannot. For example, a request addressed to an email server is allowed through; a request addressed to the corporate accounting system is denied. Usually, traffic destined for a Web server (port 80) or an email server (port 25) is granted access. Unless you specify otherwise, a firewall typically blocks all traffic addressed to other locations (i.e., servers, databases, or application servers) on the network, thus protecting those hosts against unauthorized external access. There are various firewall products but they are grouped into three major types based on their mechanisms: packet filtering, stateful inspection, and proxying. Packet filtering is a mechanism that control which packets can go to and come from a network by examining their headers. There are no content-based decisions. The decision is solely based on the packet headers which include source address, destination address, type of traffic (such as TCP, UDP, ICMP), and characteristics of the transport layer communications sessions (such as source and destination ports). Packet filters are associated with interfaces therefore; the interface that the packet comes from or will go through can be restricted. Packet filter firewall can have rules such as; letting some hosts send email via SMTP (Simple Mail Transfer Protocol) or not letting any outside host connect to an internal host using Telnet. Packet filter firewalls provide transparent security as they work at lower layers. It does not require any user knowledge or any configuration on private network hosts. Since it is easy to implement, it is widely available in many routers. However, some protocols are not well suited to packet filtering and, some security policies cannot be enforced by packet filtering. Moreover, packet filter firewalls make decisions for each network packet alone and does not examine the status of the connections that the packets belong to. Another technology, stateful inspection, evolved from the need to accommodate certain features of the TCP/IP protocol suite. In essence, stateful inspection firewalls are packet filter firewalls with the connection status awareness capability. This awareness is done by making a dynamic list of active connections between hosts, called state table. A packet that does not belong to an active connection and is not a connection request is refused by the firewall. A packet that belongs to an active connection is allowed through, bypasses the firewall rules; therefore optimizing the inspection process. Stateful inspection firewalls share the strengths and weaknesses of packet filter firewalls, but due to the state table implementation, stateful inspection firewalls are generally considered to be more secure than packet filter firewalls. A stateful inspection firewall also differs from a packet filter firewall in that stateful inspection is useful or applicable only within TCP/IP network infrastructures. Stateful inspection firewalls can accommodate other network protocols in the same manner as packet filter firewalls, but the actual stateful inspection technology is relevant only to TCP/IP [NIST02]. Lastly, proxying is a mechanism that provides all internal hosts the external (untrusted) network access while appearing that a single host accessing the outside. Since all connection to the external network be done by a single host, deep packet inspection is possible before passing packets to internal hosts. Proxying examines source address, destination address, protocol used, source port, destination port and also payload (content) of packets. Proxying allows writing complex set of rules that cannot be done in packet filtering and stateful inspection. For example, "put" commands in FTP connections with a specific host can be rejected by a rule. Proxying can also provide many forms of user authentication and allow specifying different policies for different users. However, these benefits come with a lot of process cost and it requires a configuration at internal hosts. In other words, it does not provide transparent security. Therefore, it is mostly used for only HTTP protocol.

3.5 Distributed Denial of Service Attack

DDOS attacks multiple computers and multiple internet connections are used which are distributed globally to make an attack. In this situation the victim will be flooded with the packets send from many hundreds and thousands of sources.

Distributed Denial of Service Attack is composed of four elements,

- The real attacker.
- The handlers or masters, which are compromised hosts with a special program running on them, capable of controlling multiple agents.
- The attack daemon agents or zombie hosts, who are compromised hosts that are running a special program and are responsible
for generating a stream of packets towards the intended victim. Those machines are commonly external to the victim's own network, to avoid efficient response from the victim, and external to the network of the attacker, to avoid liability if the attack is
traced back.
- A victim or target host.

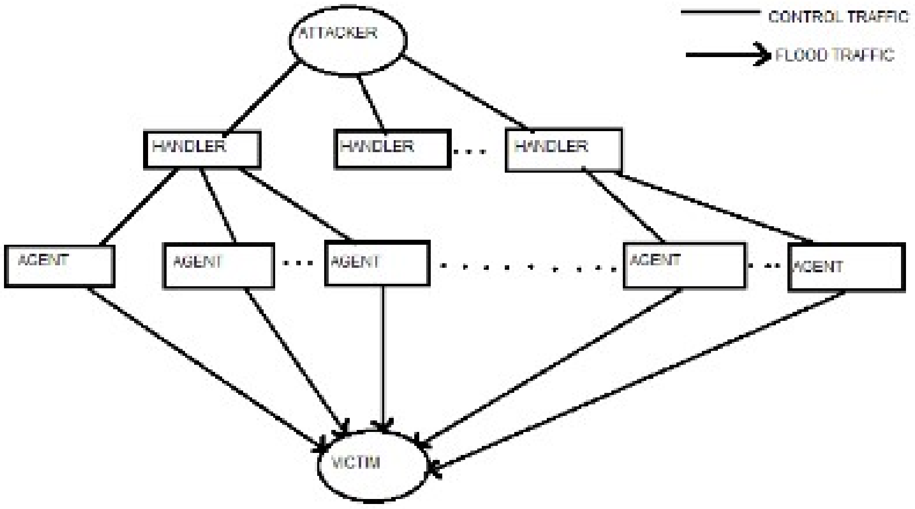


Fig 6. DDoS

DDoS attacks can be divided into three main categories:

1. Volume based attacks: -These include ICMP floods, UDP floods and other spoofed packet attacks. The main goal of the attacker is to consume the bandwidth of the victim's site. The magnitude of the attack is measured in bits per second (Bps).

2. Protocol based attacks: -These include SYN floods, fragmented packet attacks, Ping of death, Smurf attack and more. The main goal of the attacker is to consume actual server resources, such as firewall. The magnitude of the attack is measured in Packets per second.

3. Application layer-based attack: - These include attacks like Zero-day attack, Slowloris etc. the main goal of the attacker is to target the Apache, Windows or open BSD vulnerabilities and more.

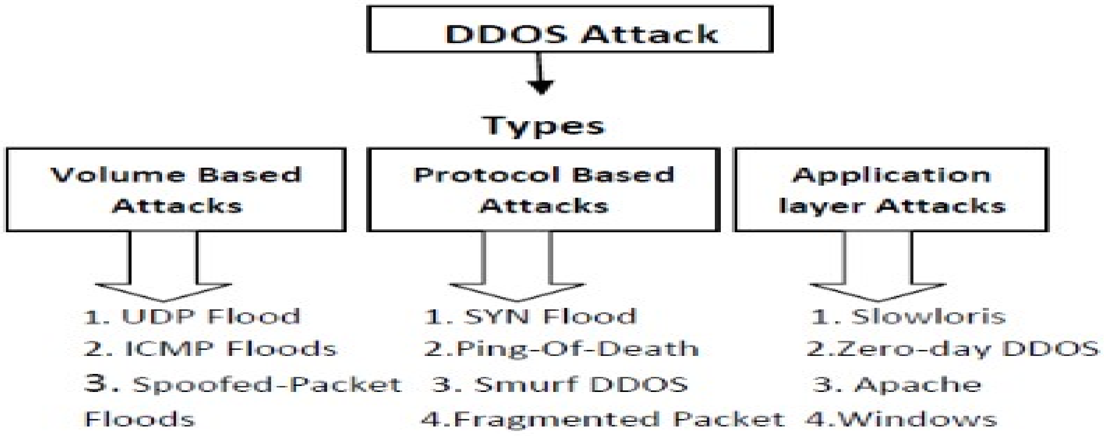


Fig 7. Classification of DDoS attack

Intrusion Detection System [11]

A second layer in the perimeter defense is intrusion detection systems (IDSs). The audits of security existed before the intrusion detection. Audit is the process of generating, storing and revising events of a system chronologically. IDS is the evolved version of the traditional audits. The term audit, in Latin “auride” (to hear), is defined as “to examine the economic management of a company in order to verify if it is adjusted to the established rules by law or custom”. Intrusion detection is the process of monitoring and searching networks of computers and systems for security policy violations [BR00]. Intrusion Detection Systems (IDSs) are software or hardware products that automate this monitoring and analysis process. An IDS inspects all inbound and outbound network activity, system logs and events, and identifies suspicious patterns or events that may indicate a network or system attack from someone attempting to break into or compromise a system. Theoretically, IDSs work like a burglar alarm, alerting security managers that an attack may be taking place so that they can respond accordingly. IDSs trigger these alerts by detecting anomalous traffic patterns or “signatures” that are characteristic of an attack. As in the physical world, our logical burglar alarm provides valuable notification that someone has managed to breach perimeter security measures, and should allow security managers to determine exactly what happened during the attack, and hopefully provide indications of how the security weakness might be addressed. IDSs have gained acceptance as a necessary addition to every organization’s security infrastructure. Since they are first put on the security market, those organizations have several compelling reasons to acquire and use IDSs. Some of them are listed below:

To prevent problematic behaviors by increasing the perceived risk of discovery and punishment for those who would attack or otherwise abuse the system

- To detect attacks and other security violations that are not prevented by other security measures
- To detect and deal with the preambles to attacks (commonly experienced as network probes and other reconnaissance activities)
- To document the existing threat to an organization
- To act as quality control tool for security design and administration, especially for large and complex enterprises
- To provide useful information about intrusions that take place, allowing detailed analysis, recovery, and correction of causative factors. There exist various IDS products in the market today. These products are categorized in several ways according to their different characteristics:

Misuse detection vs. anomaly detection: In misuse detection, the IDS analyzes the information it gathers and compares it to large database of attack signatures which causes it being also called signature-based detection. It is easy to understand the concept as it uses simple comparisons. Essentially, the IDS looks for a specific attack that has already been documented. Like a virus detection system, misuse detection software is only as good as the database of attack signatures that it uses to compare packets against. In anomaly detection, the security manager defines the baseline, or normal, state of the network's traffic load, breakdown, protocol, and typical packet size. The anomaly detector monitors network segments to compare their state to the normal baseline and looks for anomalies. Therefore, anomaly detection is as good as its baseline definition.

Network-based vs. host-based systems: In a network-based system, or NIDS, the individual packets flowing through the network are analyzed. Network-based IDSs often consist of a set of single-purpose sensors or hosts placed at various points in a network. These units monitor network traffic, performing local analysis of that traffic and reporting attacks to a central management console. NIDSs analyze traffic moving across the network in much greater detail than a firewall. Therefore, NIDSs can detect malicious packets that are designed to be overlooked by a firewall's simplistic filtering rules. NIDSs also watch for attacks that originate from within a network. That is why, they are complement for firewalls. In a host-based system or HIDS, activities on each individual computer or host are examined. An HIDS performs analysis of the local machine as it is running on. This commonly means that the HIDS software monitors log files, or other artifacts of incidents to detect that a security incident has occurred. HIDSs are not limited to log-file analyzers; they also include intra-kernel based mechanisms that detect ongoing security incidents. It is advisable to place HIDSs on all mission critical systems, even those that should not, in theory, allow external access.

Passive system vs. reactive system: In a passive system, the IDS detects a potential security breach, logs the information and signals an alert. Security manager has to examine the logs constantly and take the required measures. In a reactive system, the IDS respond to the suspicious activity by resetting the connection, by logging out the user or by reconfiguring the firewall to block network traffic from the suspected source. Intrusion detection using signature: The evolution of NIDSs started with the implementation of a non-intrusive packet monitor, called a sniffer because of its ability to “sniff” the packets on the network. Intrusion detection vendors applied the packet-monitoring concept to build systems that performed packet signature detection [NetScreen02]. Signature-based detectors analyze system activity, looking for events or sets of events that match a predefined pattern of events that describe a known attack. They compare events and packets with signatures stored in their database and find out the matching ones. The most common form of signature-based detection used in commercial products specifies each pattern of events corresponding to an attack as a separate signature.

However, there are many drawbacks to signature-based approach to intrusion detection – especially if effort is placed entirely on building up a large repository of attack signatures, without regard to how the traffic is reassembled, decoded, normalized and analyzed. It is a problem when information is transmitted over the network, the information is split into numbered TCP (Transmission Control Protocol) segments that are sent as packets. In an ideal world, the packets would be transmitted in sequence and without loss. But, unfortunately, that’s not the case. When a message is actually transmitted, the network will deliver the packets randomly (out of sequence) or as even smaller pieces of data (called fragments), which are broken down by networking devices, such as routers, to facilitate ease of transmission. Even worse, for whatever reason, packets can get “lost” or can be duplicated.

3.6 INTRUSION PREVENTION SYSTEMS

Definition of IPS varies widely because of the marketing purposes of security tool vendors. Every vendor has its own definition and so it seems that they are selling the right product; the most reliable and the most precious. Since technological developments are produced by vendors and many vendors exist in the market today, it is hard them to compromise on a definition. This confusion causes the new technology, called revolution by the vendors, be perceived as a minor improvement. Before going any further, a definition of IPS is required in order to clearly describe what we will examine. A definition used by a group of vendors that develop network-based IDSs is: “… As the name implies, intrusion prevention systems (IPSs) do not simply detect attacks as do IDSs; they actually prevent attacks from taking place or automatically block them upon detection. They enable an organization to take proactive, highly automated steps to guard against intrusions…”

An intrusion prevention system attempts to be proactive, and is designed to stop intrusions, preventing suspected system calls and events, blocking the offending traffic before it does any damage rather than simply raising an alert as, or after, the malicious payload has been delivered. It achieves this by sitting directly in-line with the system calls and network traffic. By sitting in-line, IPS, ideally, inspecting all system calls and packets going inbound or outbound. It performs a range of detection analyses. If the IPS deems the system calls unsuspicious or the packet harmless, it forwards it. End users are unaware of any effect. However, when the IPS detects suspicious system calls or packets, it can then initiate one of many response mechanisms that security manager has configured. It may restrict the system call or packet, by forwarding it normally up to a certain limit. Or, the IPS can discard it completely. Of course, an IPS must also have an extensive reporting mechanism – but this must be more than a simple log of activity. The IPS can create an alarm and transmit it to appropriate destinations. As with IDS systems, IPS products fall into two categories: Network-based IPS (NIPS) and host-based IPS (HIPS).

Network-based IPS: Network-based IPS (sometimes known as an In-line IDS or Gateway IDS (GIDS)) is a device put on the network in a critical data path that inspects all the traffic allowed through by the firewall. It could be thought of a something of a hybrid system, combining features of a standard IDS, a firewall and, sometimes, a network-based antivirus system. Those prevention products use various methods to spot trouble, such as looking for the characteristic signatures of known viruses or comparing the current traffic to a baseline of normal traffic behavior. If the devices detect anomalies, they block the traffic from continuing onto the network. Thus, they provide truly effective protection for computing resources on a large scale.

Host-based IPS: To provide truly effective protection for computing resources on a large scale, security managers cannot rely only on the detection capability of a network-based IPS. It is an essential rule of security: do not rely upon any single solution or process for protection ("defense in depth"). Thus, an additional layer of security is required, that is host-based IPS. Host-based IPSs provide protection at the end point of attacks and allows much more targeted detection and prevention of intrusions. Host-based IPSs flag intrusions by comparing the behavior of systems against expected norms. If deviations occur, the systems then have some way of blocking the procedures that are causing the anomalous behavior without affecting the machine's normal operations. The main approach in host-based IPSs is to define appropriate behaviors and then enforce those behaviors on every end-user desktop and network server across an enterprise. It's because solutions that are implemented by replacing shared libraries or analyzing system audit logs can be bypassed relatively easily. The problem is to define good or expected behavior. For example; it is an appropriate behavior that an email management software just displays the emails that the users selects, but it is not appropriate, the software immediately attempts to send email to every contact listed in that user's address book after it is displayed. Likewise, if a process originating from a web browser, mail software or Microsoft Office program group attempts to read, write or modify files in its program folder or temporary folder is appropriate but, it is not appropriate if it writes to Windows system files. It is by looking at system and application behavior in this way and defining which actions are legitimate and which actions are suspect that pioneering intrusion prevention technologies can preemptively neutralize an errant system action when it attempts to do something that is outside the realm of expected behavior.

3.7 NETWORK FORENSICS USING SNORT

Snort is a NIDS (network intrusion detection system) designed to capture live network traffic or playback precaptured network traffic for advance intrusion analysis [12]. The precaptured network traffic should be saved as a “de facto” standard. The “de facto” standard for network data is the libpcap library format known as *pcap* (for UNIX/Linux-based operating systems [OSes]). For Microsoft Windows-based OSes, the library format is known as *WinPcap,* but it is the same format as the UNIX/Linux-based pcap.

For detecting the network attacks, Network forensic analysis plays an important role and the architecture on which network forensic analysis depends upon tools used for network forensic analysis. Generally, we use the Snort tool for the detection of malicious packet. Snort, a free open - source multiplatform product, can be configured to run in four modes. the First mode is sniffer, function as a packer sniffer that reads the packets off the network. The second mode, packets logger, can be configured to log the packets to disk. The third mode is,NIDS allow snort to analyze decoded network traffic against predefined preprocessors and rules and performs several different actions if a match is found. The fourth move is inline, allows snort to obtain packets and drops or pass those packets based on Snort inline specific rule types.

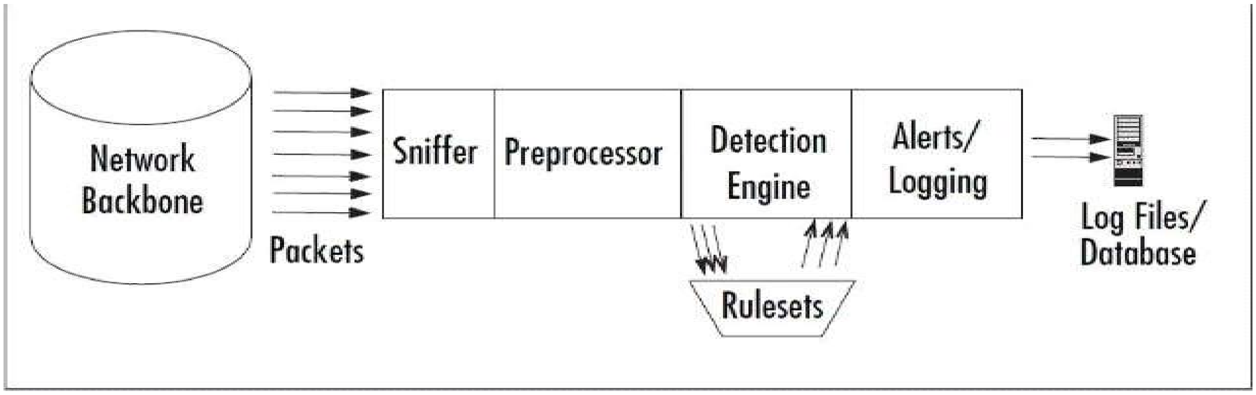


Fig 8. SNORT ARCHITECTURE

The rule header is the portion of the rule that identifies how to match the traffic which is based on the following criteria:

- Action (alert, drop, pass, and so on)
- Protocol (TCP, UDP, ICMP, IP)
- Source IP
- Source port
- Direction operator (the IP address and port numbers on the left side of the direction operator `->`) is considered to be the source. There is also a bidirectional operator, which is indicated with a `<>` symbol which tells Snort to consider the address/port pairs in either the source or destination orientation. Also, note that there is no `<-` direction operator).
- Destination IP
- Destination port

Every rule header must specify these parameters.

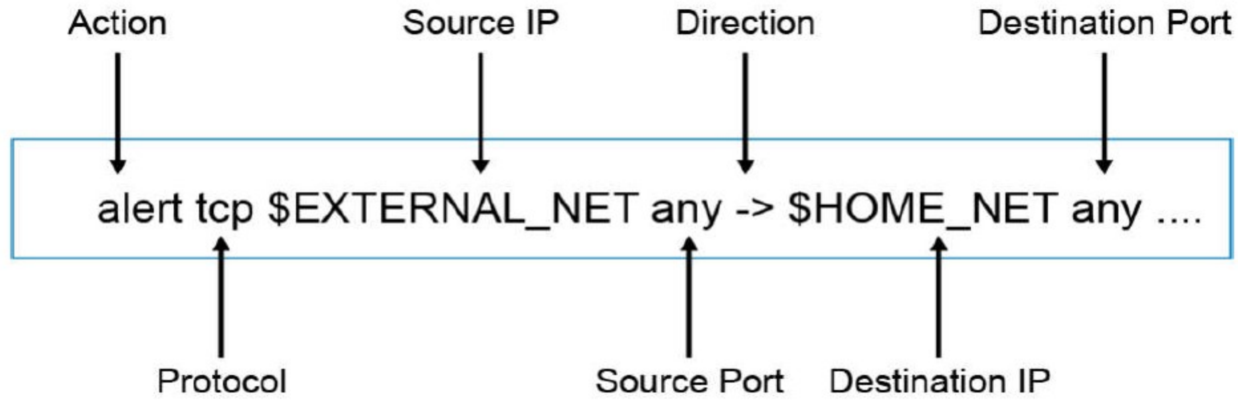
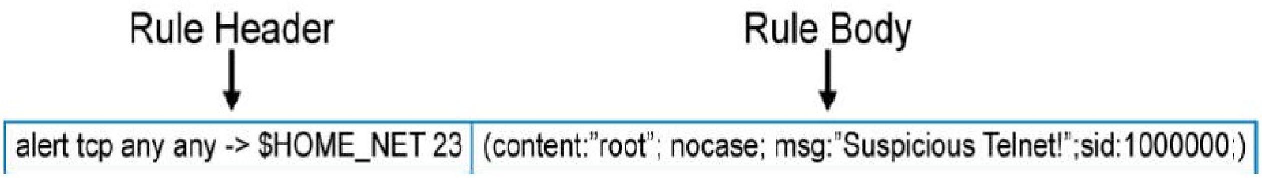


Fig 9. Rule Header

The rule body is where the real power of the Cisco Snort rule language comes into play. In the rule body, you can drill into a packet and get to the content that actually signals malicious or suspicious activity.



```
alert tcp any any -> $HOME_NET 23 (content:"root"; nocase; msg:"Suspicious Telnet!";sid:1000000;)
```

Fig 10. Categorization of Rule header and Rule body.

3.8 NETWORK FORENSICS USING WIRESHARK

Wireshark, is a network analysis tool that captures packets in real time and displays them in a readable format. Wireshark provides a variety of options such as filters, color-coding, and other features that let you analyze network traffic and inspect individual packets. It is most often used for network troubleshooting, analysis, software and communications protocol development, and network forensics.

Wireshark is a robust program that allows for the following:

- Using filters can greatly assist in narrowing data, as Wireshark tends to generate a lot of data that may not all be useful.
- Wireshark can read live data from multiple network types, including Ethernet and IEEE 802.11.
- Wireshark can capture raw USB traffic.
- Wireshark has a GUI for analysis; however, it also has a command line version called TShark.
- Data can be captured directly from a live network or read from already-captured packets.

During the incident investigation process, a typical task for an analyst is to perform packet capture and packet analysis. Wireshark is one of the common tools that is used to perform packet capture and analysis. Wireshark is available for windows ,linux etc. After capturing the traffic it will be generated into a pcap file.

4.SOFTWARE REQUIREMENT & SPECIFICATION

4.1 Software Required:

1. WireShark
2. Snort
3. Anaconda IDE

4.2 Hardware Required:

As the project does not involve any database, its hardware requirements are minimal. Any System with intel i3 or above processor, 4GB RAM, 500GB Hard Disk, and Wireless Network is sufficient. Its network based software so computers connected with any kind of mode (wireless, LAN connected etc) will suit its requirements.

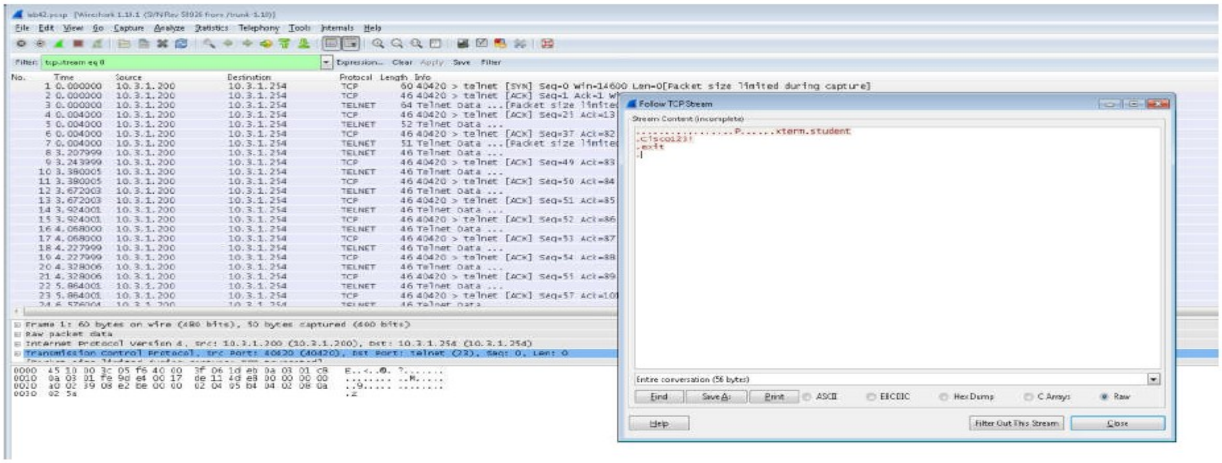
4.3Software Analysis Report:

About : Wireshark

Wireshark, is a network analysis tool that captures packets in real time and displays them in a readable format. Wireshark provides a variety of options such as filters, color-coding, and other features that let you analyze network traffic and inspect individual packets. It is most often used for network troubleshooting, analysis, software and communications protocol development, and network forensics.
Wireshark is a robust program that allows for the following:

- Using filters can greatly assist in narrowing data, as Wireshark tends to generate a lot of data that may not all be useful.
- Wireshark can read live data from multiple network types, including Ethernet and IEEE 802.11.
- Wireshark can capture raw USB traffic.
- Wireshark has a GUI for analysis; however, it also has a command line version called TShark.
- Data can be captured directly from a live network or read from already-captured packets.

During the incident investigation process, a typical task for an analyst is to perform packet capture and packet analysis. Wireshark is one of the common tools that is used to perform packet capture and analysis. Wireshark is available for windows ,linux etc. After capturing the traffic it will be generated into a pcap file.


lab42.pcap [Wireshark 1.10.1 (SVN Rev 50926 from /trunk-1.10)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Expression... Clear Apply Save Filter
Follow TCP Stream
Find Save As Print
○ ASCII ○ EBCDIC ○ Hex Dump ○ C Arrays ● Raw
Help Filter Out This Stream Close

Filter: tcp.stream eq 0

```
No.  Time       Source        Destination   Protocol  Length Info
   1 0.000000   10.3.1.200    10.3.1.254    TCP       60 40420 > telnet [SYN] Seq=0 Win=14600 Len=0[Packet size limited during capture]
   2 0.000000   10.3.1.200    10.3.1.254    TCP       46 40420 > telnet [ACK] Seq=1 Ack=1 Wi
   3 0.000000   10.3.1.200    10.3.1.254    TELNET    64 Telnet Data ...[Packet size limite
   4 0.004000   10.3.1.200    10.3.1.254    TCP       46 40420 > telnet [ACK] Seq=21 Ack=13
   5 0.004000   10.3.1.200    10.3.1.254    TELNET    52 Telnet Data ...
   6 0.004000   10.3.1.200    10.3.1.254    TCP       46 40420 > telnet [ACK] Seq=37 Ack=82
   7 0.004000   10.3.1.200    10.3.1.254    TELNET    51 Telnet Data ...[Packet size limite
   8 3.207999   10.3.1.200    10.3.1.254    TELNET    46 Telnet Data ...
   9 3.243999   10.3.1.200    10.3.1.254    TCP       46 40420 > telnet [ACK] Seq=49 Ack=83
  10 3.380005   10.3.1.200    10.3.1.254    TELNET    46 Telnet Data ...
  11 3.380005   10.3.1.200    10.3.1.254    TCP       46 40420 > telnet [ACK] Seq=50 Ack=84
  12 3.672003   10.3.1.200    10.3.1.254    TELNET    46 Telnet Data ...
  13 3.672003   10.3.1.200    10.3.1.254    TCP       46 40420 > telnet [ACK] Seq=51 Ack=85
  14 3.924001   10.3.1.200    10.3.1.254    TELNET    46 Telnet Data ...
  15 3.924001   10.3.1.200    10.3.1.254    TCP       46 40420 > telnet [ACK] Seq=52 Ack=86
  16 4.068000   10.3.1.200    10.3.1.254    TELNET    46 Telnet Data ...
  17 4.068000   10.3.1.200    10.3.1.254    TCP       46 40420 > telnet [ACK] Seq=53 Ack=87
  18 4.227999   10.3.1.200    10.3.1.254    TELNET    46 Telnet Data ...
  19 4.227999   10.3.1.200    10.3.1.254    TCP       46 40420 > telnet [ACK] Seq=54 Ack=88
  20 4.328006   10.3.1.200    10.3.1.254    TELNET    46 Telnet Data ...
  21 4.328006   10.3.1.200    10.3.1.254    TCP       46 40420 > telnet [ACK] Seq=55 Ack=89
  22 5.864001   10.3.1.200    10.3.1.254    TELNET    46 Telnet Data ...
  23 5.864001   10.3.1.200    10.3.1.254    TCP       46 40420 > telnet [ACK] Seq=57 Ack=10
  24 6.576004   10.3.1.200    10.3.1.254    TELNET    46 Telnet Data

⊞ Frame 1: 60 bytes on wire (480 bits), 50 bytes captured (400 bits)
⊞ Raw packet data
⊞ Internet Protocol Version 4, Src: 10.3.1.200 (10.3.1.200), Dst: 10.3.1.254 (10.3.1.254)
⊞ Transmission Control Protocol, Src Port: 40420 (40420), Dst Port: telnet (23), Seq: 0, Len: 0

0000  45 10 00 3c 05 f6 40 00  3f 06 1d eb 0a 03 01 c8   E..<..@. ?.......
0010  0a 03 01 fe 9d e4 00 17  de 11 4d e8 00 00 00 00   ........ ..M.....
0020  a0 02 39 08 e2 be 00 00  02 04 05 b4 04 02 08 0a   ..9..... ........
0030  02 5a                                              .Z
```

Stream Content (incomplete)

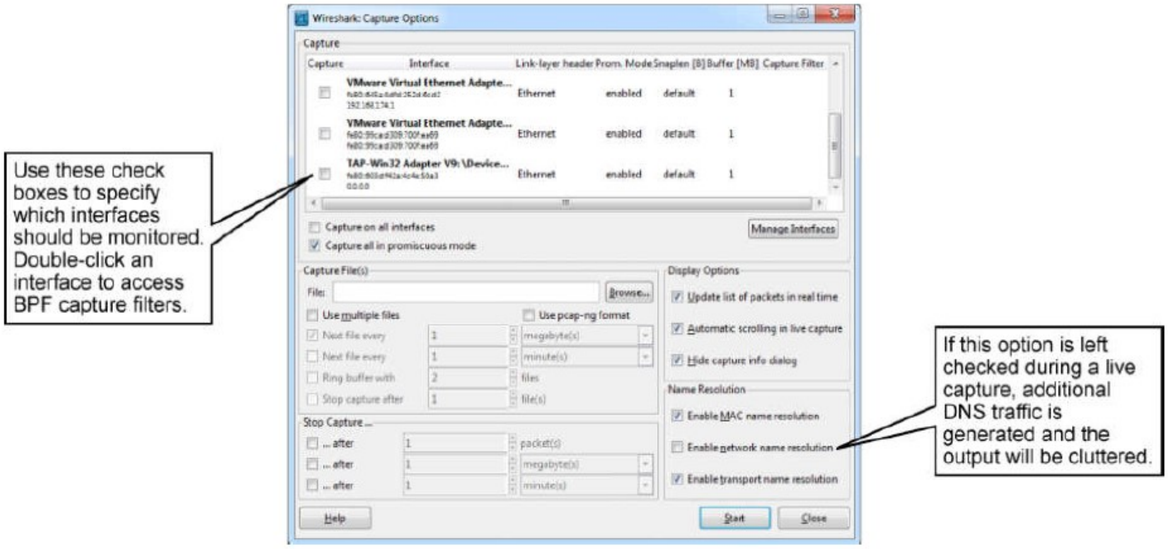
```
..................P......xterm.student
.cisco123!
.exit
.|
```

Entire conversation (56 bytes)

Fig 11. Using Wireshark to examine PCAP file

This section provides a high-level overview of Wireshark. Throughout the different labs in the course, you will use Wireshark to perform packet capture and to analyze the PCAPs.

Clicking **Capture Options** from the main screen opens the **Capture Options** window. The **Capture Options** window allows you to select an interface if there are multiple interfaces, define filenames for output, set display options, end capture options, and define name-resolution options. If you are capturing traffic live, it is strongly recommended that you disable network name resolution by unchecking the **Enable Network Name Resolution** check box.



Wireshark: Capture Options

Capture

Capture	Interface	Link-layer header	Prom. Mode	Snaplen [B]	Buffer [MB]	Capture Filter
□	VMware Virtual Ethernet Adapte... fe80::640a:6a9d:352d:4cd2 192.168.174.1	Ethernet	enabled	default	1	
□	VMware Virtual Ethernet Adapte... fe80::99ca:d309:700f:ea69 fe80::99ca:d309:700f:ea69	Ethernet	enabled	default	1	
□	TAP-Win32 Adapter V9: \Device... fe80::603d:f42a:4c4e:50a3 0.0.0.0	Ethernet	enabled	default	1	

□ Capture on all interfaces
☑ Capture all in promiscuous mode

Capture File(s)

File:

□ Use multiple files   □ Use pcap-ng format
☑ Next file every 1 megabyte(s)
□ Next file every 1 minute(s)
□ Ring buffer with 2 files
□ Stop capture after 1 file(s)

Stop Capture ...

□ ... after 1 packet(s)
□ ... after 1 megabyte(s)
□ ... after 1 minute(s)

Display Options

☑ Update list of packets in real time
☑ Automatic scrolling in live capture
☑ Hide capture info dialog

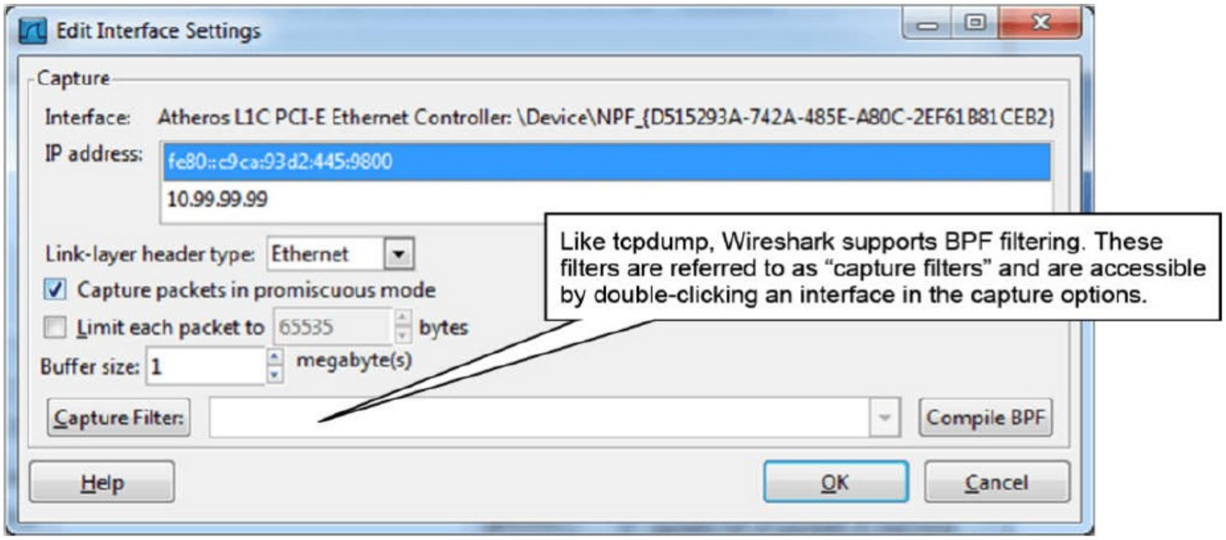
Name Resolution

☑ Enable MAC name resolution
□ Enable network name resolution
☑ Enable transport name resolution

Use these check boxes to specify which interfaces should be monitored. Double-click an interface to access BPF capture filters.

If this option is left checked during a live capture, additional DNS traffic is generated and the output will be cluttered.

Fig 12. Wireshark Capture Option



Edit Interface Settings

Capture

Interface: Atheros L1C PCI-E Ethernet Controller: \Device\NPF_{D515293A-742A-485E-A80C-2EF61B81CEB2}

IP address: fe80::c9ca:93d2:445:9800
10.99.99.99

Link-layer header type: Ethernet

☑ Capture packets in promiscuous mode

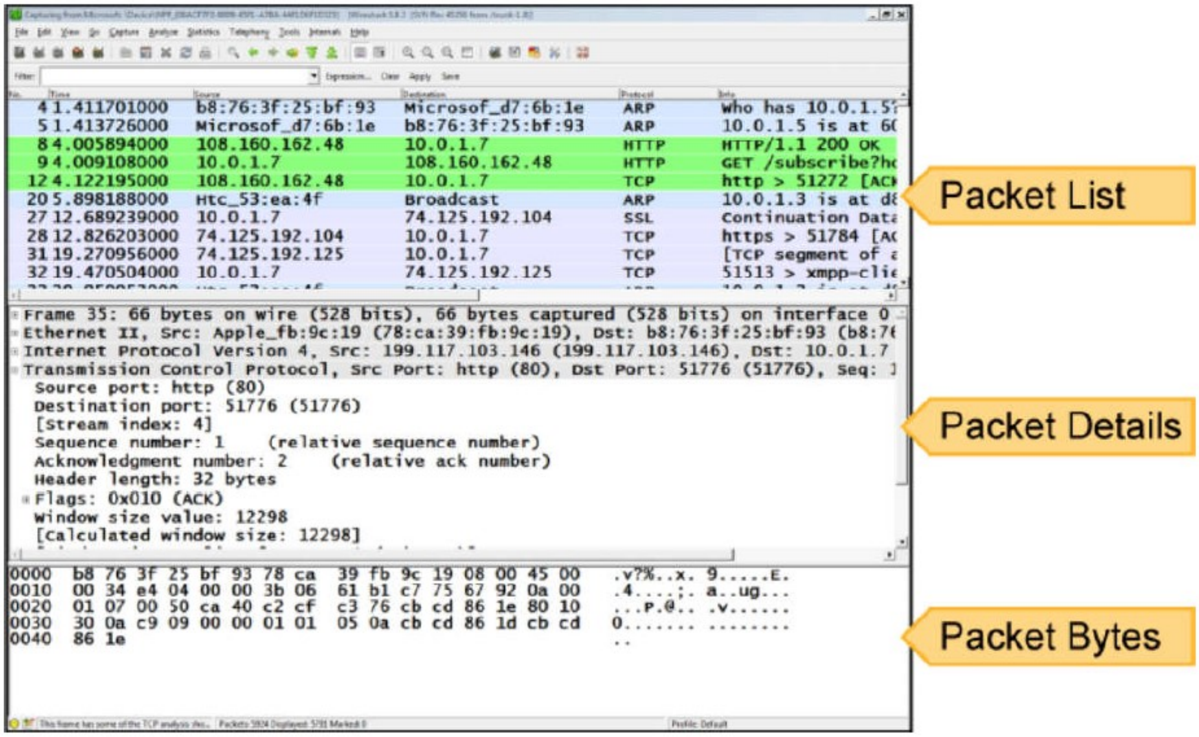
□ Limit each packet to 65535 bytes

Buffer size: 1 megabyte(s)

Capture Filter:

Like tcpdump, Wireshark supports BPF filtering. These filters are referred to as "capture filters" and are accessible by double-clicking an interface in the capture options.

Fig 13. Edit Interface Settings in WireShark



```
  4 1.411701000  b8:76:3f:25:bf:93   Microsof_d7:6b:1e   ARP   Who has 10.0.1.5?
  5 1.413726000  Microsof_d7:6b:1e   b8:76:3f:25:bf:93   ARP   10.0.1.5 is at 6(
  8 4.005894000  108.160.162.48      10.0.1.7            HTTP  HTTP/1.1 200 OK
  9 4.009108000  10.0.1.7            108.160.162.48      HTTP  GET /subscribe?h(
 12 4.122195000  108.160.162.48      10.0.1.7            TCP   http > 51272 [ACK
 20 5.898188000  Htc_53:ea:4f        Broadcast           ARP   10.0.1.3 is at d8
 27 12.689239000 10.0.1.7            74.125.192.104      SSL   Continuation Data
 28 12.826203000 74.125.192.104      10.0.1.7            TCP   https > 51784 [AC
 31 19.270956000 74.125.192.125      10.0.1.7            TCP   [TCP segment of a
 32 19.470504000 10.0.1.7            74.125.192.125      TCP   51513 > xmpp-clie
```

```
Frame 35: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
Ethernet II, Src: Apple_fb:9c:19 (78:ca:39:fb:9c:19), Dst: b8:76:3f:25:bf:93 (b8:76
Internet Protocol Version 4, Src: 199.117.103.146 (199.117.103.146), Dst: 10.0.1.7
Transmission Control Protocol, Src Port: http (80), Dst Port: 51776 (51776), Seq: 1
  Source port: http (80)
  Destination port: 51776 (51776)
  [Stream index: 4]
  Sequence number: 1    (relative sequence number)
  Acknowledgment number: 2    (relative ack number)
  Header length: 32 bytes
 Flags: 0x010 (ACK)
  Window size value: 12298
  [Calculated window size: 12298]
```

```
0000   b8 76 3f 25 bf 93 78 ca  39 fb 9c 19 08 00 45 00   .v?%..x. 9.....E.
0010   00 34 e4 04 00 00 3b 06  61 b1 c7 75 67 92 0a 00   .4....;. a..ug...
0020   01 07 00 50 ca 40 c2 cf  c3 76 cb cd 86 1e 80 10   ...P.@.. .v......
0030   30 0a c9 09 00 00 01 01  05 0a cb cd 86 1d cb cd   0....... ........
0040   86 1e                                              ..
```

Fig 14. Main interface of Wireshark

The main interface of Wireshark consists of three components:

- The packet list shows a complete list of packets within the current capture. Information about each packet is presented in customizable columns. By default, this information includes the packet number, time stamp, source address, destination address, protocol, and a summary field of protocol-specific information.
- The packet details list shows detailed information about the highlighted packet. Protocols within the packet are presented in expandable panes, with each field enumerated and explained. Some basic analysis is performed, such as translating port numbers to more human-readable names or displaying the human-readable flags in the **Flags** field.
- The packet bytes pane shows the raw bytes of the highlighted packet, starting at the link-level header. The output is divided into three columns: offset, hexadecimal representation, and ASCII representation.

In the main interface, analysts can quickly move around the capture to inspect packets of interest. Clicking a packet in the top third of the window, the packet list, alters the other two panes to show the details and bytes of the highlighted packet.

About Snort

Snort is a NIDS (network intrusion detection system) designed to capture live network traffic or playback precaptured network traffic for advance intrusion analysis [12]. The precaptured network traffic should be saved as a "de facto" standard. The "de facto" standard for network data is the libpcap library format known as *pcap* (for UNIX/Linux-based operating systems [OSes]). For Microsoft Windows-based OSes, the library format is known as *WinPcap,* but it is the same format as the UNIX/Linux-based pcap.

For detecting the network attacks, Network forensic analysis plays an important role and the architecture on which network forensic analysis depends upon tools used for network forensic analysis. Generally we use the Snort tool for the detection of malicious packet. Snort, a free open - source multiplatform product, can be configured to run in four modes .the First mode is sniffer ,function as a packer sniffer that reads the packets off the network. The second mode, packets logger, can be configured to log the packets to disk .The third mode is,NIDS allow snort to analyze decoded network traffic against predefined preprocessors and rules and performs several different actions if a match is found. The fourth move is inline, allows snort to obtain packets and drops or pass those packetsbased on Snort inline specific rule types.

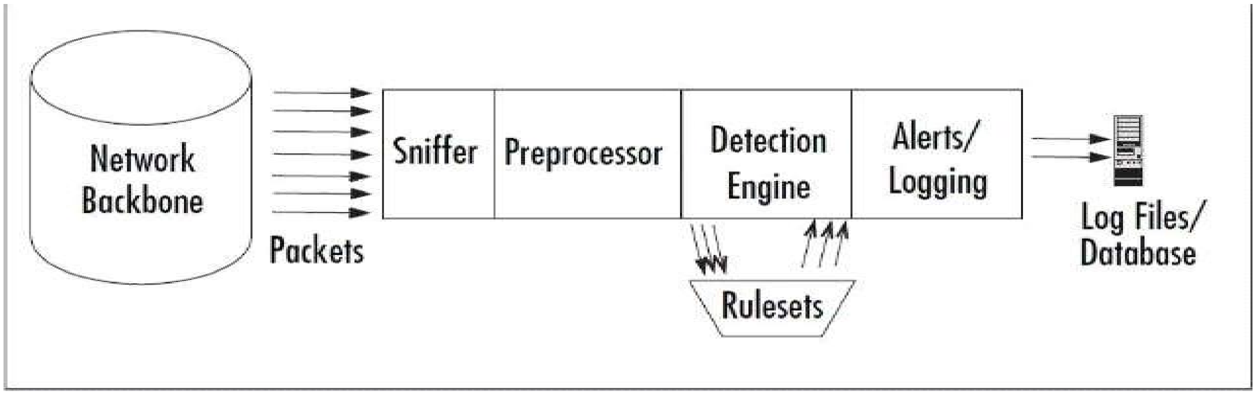


Fig 8. SNORT ARCHITECTURE

5.PLANNING

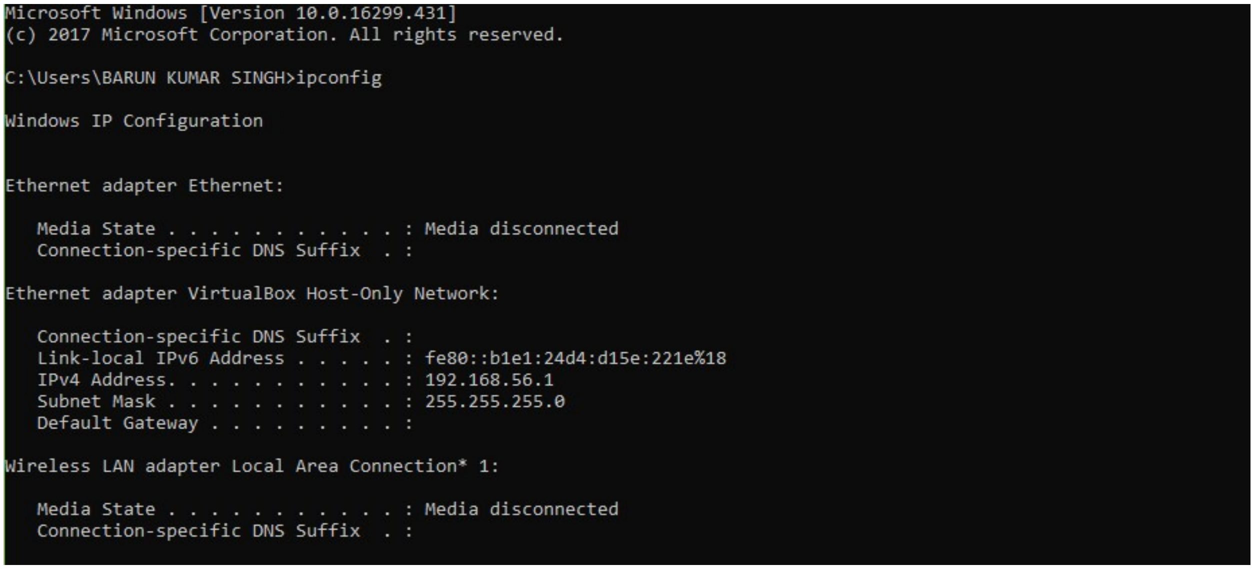
The Proposed Plan for Network Forensic Analysis

1. Finding suitable papers and documents related to network forensics analysis.
2. Read about various network based attacks and tools required to simulate them.
3. Analyzing data packets using Wireshark for network forensic analysis.
4. Detection of various attacks using Wireshark and Snort.
5. Use Snort for Intrusion prevention system.
6. Provide detailed log analysis generated from compromised system.
7. List observations from log analysis.
8. conclusion

6.DESIGN

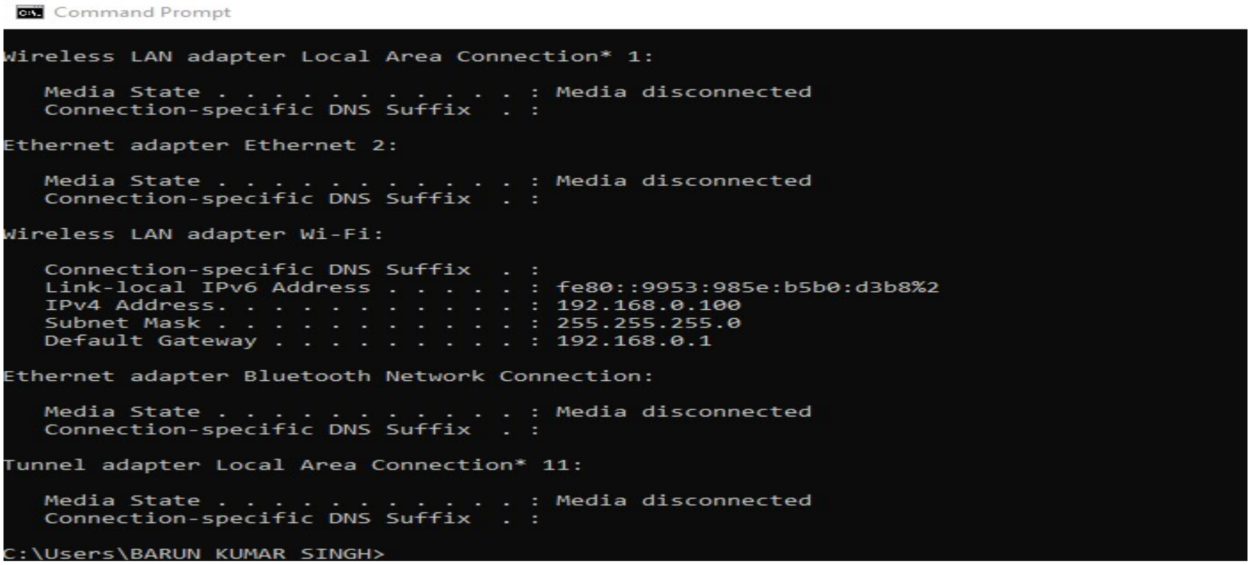
6.1 Examine the network configuration

Enter the **ipconfig** command and review its output.



```
Microsoft Windows [Version 10.0.16299.431]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\BARUN KUMAR SINGH>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Ethernet adapter VirtualBox Host-Only Network:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::b1e1:24d4:d15e:221e%18
   IPv4 Address. . . . . . . . . . . : 192.168.56.1
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Wireless LAN adapter Local Area Connection* 1:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
```

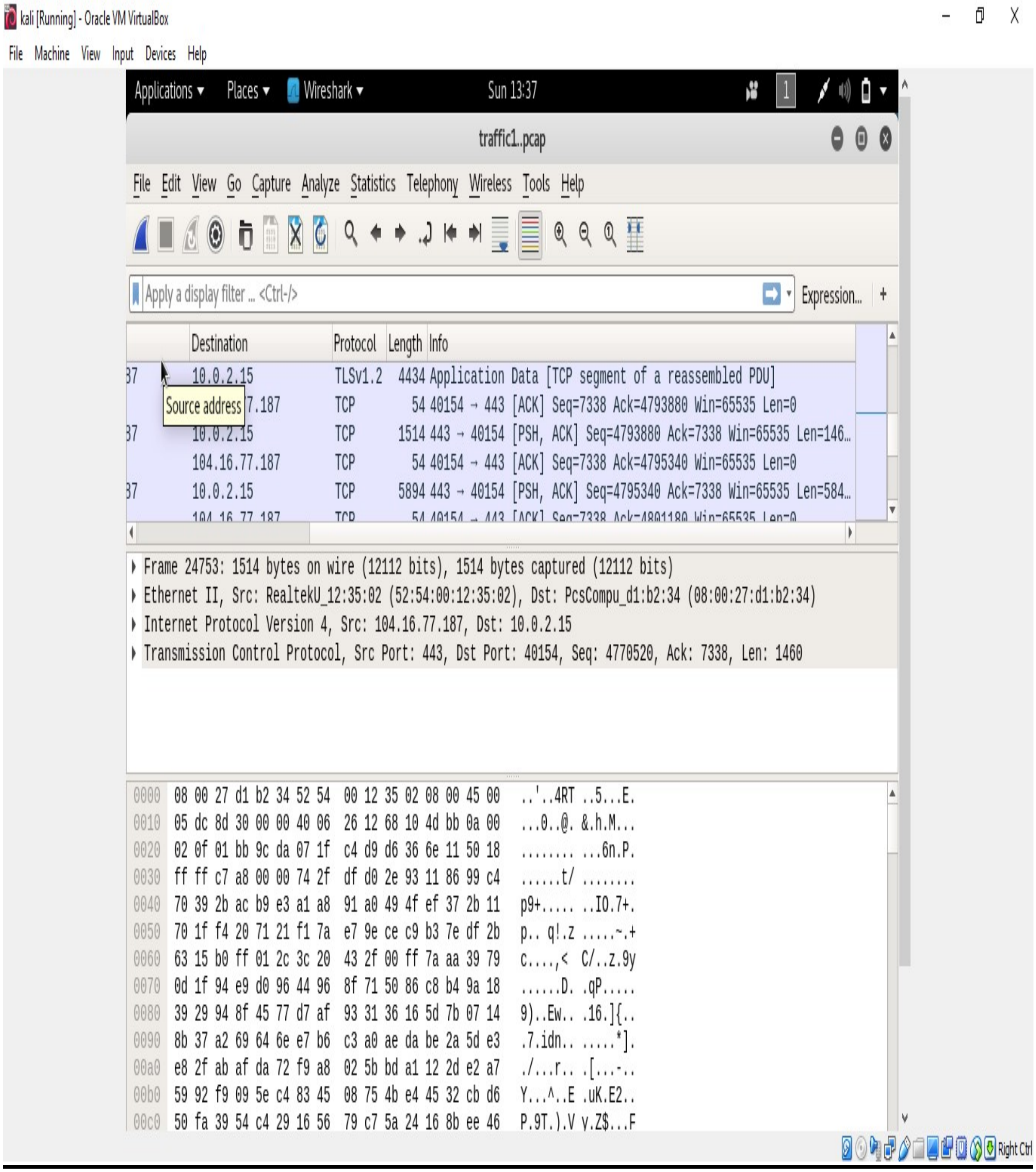
Fig 15. Examine of ipconfig



```
Wireless LAN adapter Local Area Connection* 1:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Ethernet adapter Ethernet 2:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::9953:985e:b5b0:d3b8%2
   IPv4 Address. . . . . . . . . . . : 192.168.0.100
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.0.1

Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Tunnel adapter Local Area Connection* 11:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Users\BARUN KUMAR SINGH>
```

Fig 16. Examine of ip configuration

6.2 Generate and capture local LAN traffic



kali [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications ▾ Places ▾ Wireshark ▾
Sun 13:37
traffic1..pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
Source address
Right Ctrl

```
      Destination      Protocol Length Info
37    10.0.2.15        TLSv1.2   4434 Application Data [TCP segment of a reassembled PDU]
      7.187            TCP         54 40154 → 443 [ACK] Seq=7338 Ack=4793880 Win=65535 Len=0
37    10.0.2.15        TCP       1514 443 → 40154 [PSH, ACK] Seq=4793880 Ack=7338 Win=65535 Len=146…
      104.16.77.187    TCP         54 40154 → 443 [ACK] Seq=7338 Ack=4795340 Win=65535 Len=0
37    10.0.2.15        TCP       5894 443 → 40154 [PSH, ACK] Seq=4795340 Ack=7338 Win=65535 Len=584…
      104.16.77.187    TCP         54 40154 → 443 [ACK] Seq=7338 Ack=4801180 Win=65535 Len=0
```

```
▸ Frame 24753: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits)
▸ Ethernet II, Src: RealtekU_12:35:02 (52:54:00:12:35:02), Dst: PcsCompu_d1:b2:34 (08:00:27:d1:b2:34)
▸ Internet Protocol Version 4, Src: 104.16.77.187, Dst: 10.0.2.15
▸ Transmission Control Protocol, Src Port: 443, Dst Port: 40154, Seq: 4770520, Ack: 7338, Len: 1460
```

```
0000  08 00 27 d1 b2 34 52 54  00 12 35 02 08 00 45 00   ..'..4RT ..5...E.
0010  05 dc 8d 30 00 00 40 06  26 12 68 10 4d bb 0a 00   ...0..@. &.h.M...
0020  02 0f 01 bb 9c da 07 1f  c4 d9 d6 36 6e 11 50 18   ........ ...6n.P.
0030  ff ff c7 a8 00 00 74 2f  df d0 2e 93 11 86 99 c4   ......t/ ........
0040  70 39 2b ac b9 e3 a1 a8  91 a0 49 4f ef 37 2b 11   p9+..... ..IO.7+.
0050  70 1f f4 20 71 21 f1 7a  e7 9e ce c9 b3 7e df 2b   p.. q!.z .....~.+
0060  63 15 b0 ff 01 2c 3c 20  43 2f 00 ff 7a aa 39 79   c....,<  C/..z.9y
0070  0d 1f 94 e9 d0 96 44 96  8f 71 50 86 c8 b4 9a 18   ......D. .qP.....
0080  39 29 94 8f 45 77 d7 af  93 31 36 16 5d 7b 07 14   9)..Ew.. .16.]{..
0090  8b 37 a2 69 64 6e e7 b6  c3 a0 ae da be 2a 5d e3   .7.idn.. .....*].
00a0  e8 2f ab af da 72 f9 a8  02 5b bd a1 12 2d e2 a7   ./...r.. .[...-..
00b0  59 92 f9 09 5e c4 83 45  08 75 4b e4 45 32 cb d6   Y...^..E .uK.E2..
00c0  50 fa 39 54 c4 29 16 56  79 c7 5a 24 16 8b ee 46   P.9T.).V y.Z$...F
```

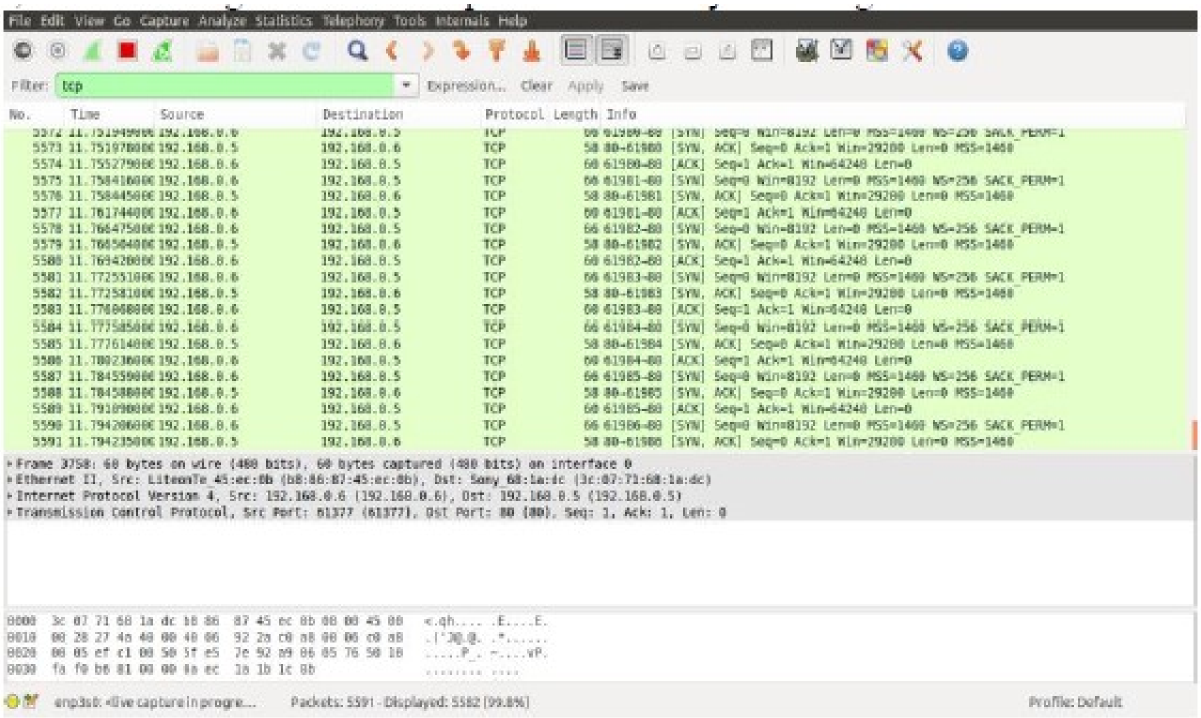
Fig 17. Generating Local Lan Traffic Using Wireshark

6.3 EXPERIMENTAL SETUP

A suitable test environment was created for carrying out the attack and testing its attributes. Ubuntu 15.10 was used as the testing OS. An Apache server httpd version 2.4.20 was installed in one of the systems with a webpage hosted in the same. Any host connected to this network would be able to request and view this webpage. We further installed wireshark on the victim system for monitoring the network attack traffic. Different tools were used for carrying out the different Dos/DDos attack.

A. TCP SYN Flood Attack

The attack was made by using the tool Hping. The victim's machine was flooded using the tool by running the following Hping command from attacker's system: **# hping3 --flood –S –p 80 192.168.0.5 --flood** flag sends the packet at a fast rate **-S** flag sets the SYN flag on in TCP mode **-p** 80 sends the packet to port 80 on victim's machine On victim machine, the following traffic was captured and analysed using Wireshark.


File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Expression... Clear Apply Save

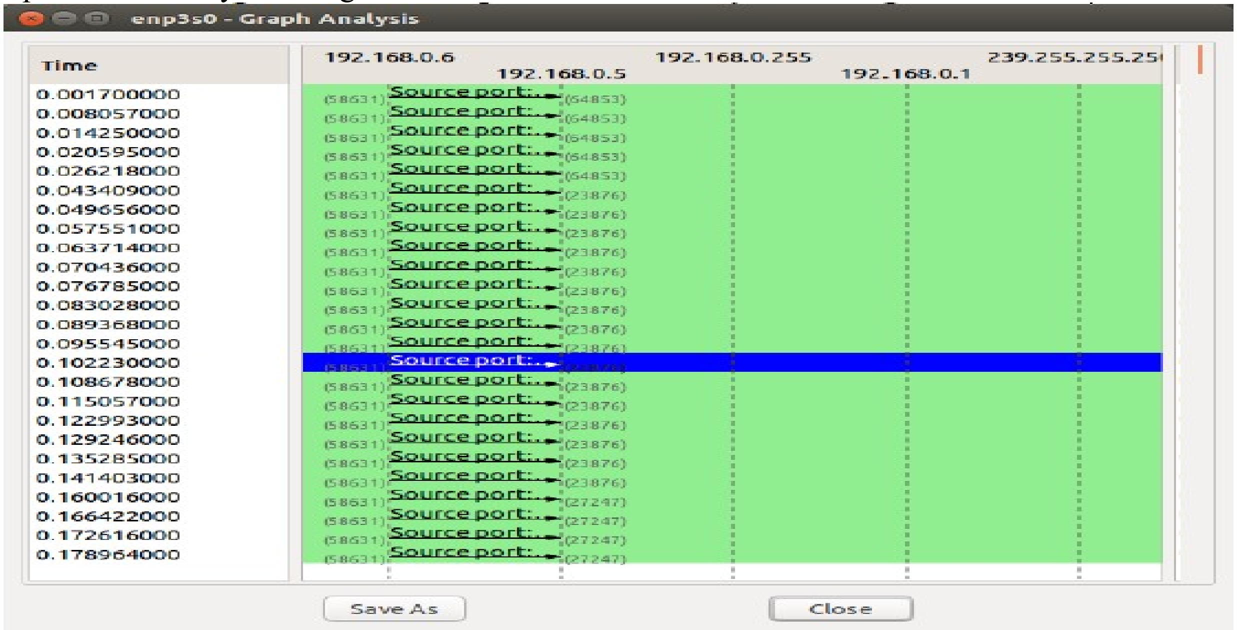
Filter: tcp

```
No.   Time         Source        Destination   Protocol Length Info
 5572 11.75194900  192.168.0.6   192.168.0.5   TCP      66 61980→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
 5573 11.75197800  192.168.0.5   192.168.0.6   TCP      58 80→61980 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
 5574 11.75527900  192.168.0.6   192.168.0.5   TCP      60 61980→80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
 5575 11.75841600  192.168.0.6   192.168.0.5   TCP      66 61981→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
 5576 11.75844500  192.168.0.5   192.168.0.6   TCP      58 80→61981 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
 5577 11.76174400  192.168.0.6   192.168.0.5   TCP      60 61981→80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
 5578 11.76647500  192.168.0.6   192.168.0.5   TCP      66 61982→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
 5579 11.76650400  192.168.0.5   192.168.0.6   TCP      58 80→61982 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
 5580 11.76942000  192.168.0.6   192.168.0.5   TCP      60 61982→80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
 5581 11.77255100  192.168.0.6   192.168.0.5   TCP      66 61983→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
 5582 11.77258100  192.168.0.5   192.168.0.6   TCP      58 80→61983 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
 5583 11.77606800  192.168.0.6   192.168.0.5   TCP      60 61983→80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
 5584 11.77758500  192.168.0.6   192.168.0.5   TCP      66 61984→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
 5585 11.77761400  192.168.0.5   192.168.0.6   TCP      58 80→61984 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
 5586 11.78023600  192.168.0.6   192.168.0.5   TCP      60 61984→80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
 5587 11.78455900  192.168.0.6   192.168.0.5   TCP      66 61985→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
 5588 11.78458800  192.168.0.5   192.168.0.6   TCP      58 80→61985 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
 5589 11.79109000  192.168.0.6   192.168.0.5   TCP      60 61985→80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
 5590 11.79420600  192.168.0.6   192.168.0.5   TCP      66 61986→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
 5591 11.79423500  192.168.0.5   192.168.0.6   TCP      58 80→61986 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460

▸ Frame 3758: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
▸ Ethernet II, Src: LiteonTe_45:ec:0b (b8:86:87:45:ec:0b), Dst: Sony_68:1a:dc (3c:07:71:68:1a:dc)
▸ Internet Protocol Version 4, Src: 192.168.0.6 (192.168.0.6), Dst: 192.168.0.5 (192.168.0.5)
▸ Transmission Control Protocol, Src Port: 61377 (61377), Dst Port: 80 (80), Seq: 1, Ack: 1, Len: 0

0000  3c 07 71 68 1a dc b8 86  87 45 ec 0b 08 00 45 00   <.qh.... .E....E.
0010  00 28 27 4a 40 00 40 06  92 2a c0 a8 00 06 c0 a8   .('J@.@. .*......
0020  00 05 ef c1 00 50 5f e5  7e 92 a9 86 05 76 50 10   .....P_. ~....vP.
0030  fa f0 b6 81 00 00 8a ec  1a 1b 1c 8b               ........ ....

enp3s0: <live capture in progre...    Packets: 5591 · Displayed: 5582 (99.8%)    Profile: Default
```

Fig 18. Analyzing TCP flood Attack using Wireshark

B. UDP Flood Attack

The attack was made by flooding the victim's machine with udp packets using Hping tool with the following command: # **hping3 –p 80 –i u1000 --udp 192.168.0.5 -p** 80 sends the packet to port 80 on victim's machine (192.168.0.5) **-i** u1000 sets the interval between packets as 100 packets per second. **--udp** flag sets the udp mode On victim machine, the following traffic was captured and analysed using Wireshark.

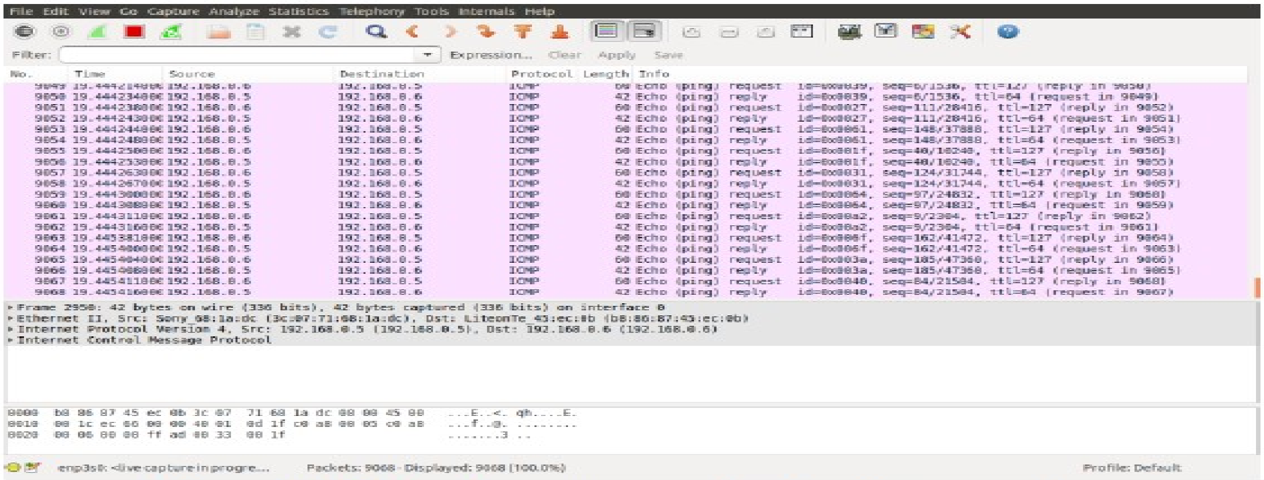


```
Time            192.168.0.6          192.168.0.255          239.255.255.25
                       192.168.0.5          192.168.0.1
0.001700000     (58631) Source port:... (54853)
0.008057000     (58631) Source port:... (54853)
0.014250000     (58631) Source port:... (54853)
0.020595000     (58631) Source port:... (54853)
0.026218000     (58631) Source port:... (54853)
0.043409000     (58631) Source port:... (23876)
0.049656000     (58631) Source port:... (23876)
0.057551000     (58631) Source port:... (23876)
0.063714000     (58631) Source port:... (23876)
0.070436000     (58631) Source port:... (23876)
0.076785000     (58631) Source port:... (23876)
0.083028000     (58631) Source port:... (23876)
0.089368000     (58631) Source port:... (23876)
0.095545000     (58631) Source port:... (23876)
0.102230000     (58631) Source port:... (23876)
0.108678000     (58631) Source port:... (23876)
0.115057000     (58631) Source port:... (23876)
0.122993000     (58631) Source port:... (23876)
0.129246000     (58631) Source port:... (23876)
0.135285000     (58631) Source port:... (23876)
0.141403000     (58631) Source port:... (23876)
0.160016000     (58631) Source port:... (27247)
0.166422000     (58631) Source port:... (27247)
0.172616000     (58631) Source port:... (27247)
0.178964000     (58631) Source port:... (27247)
```

Fig 19. UDP Flood Attack

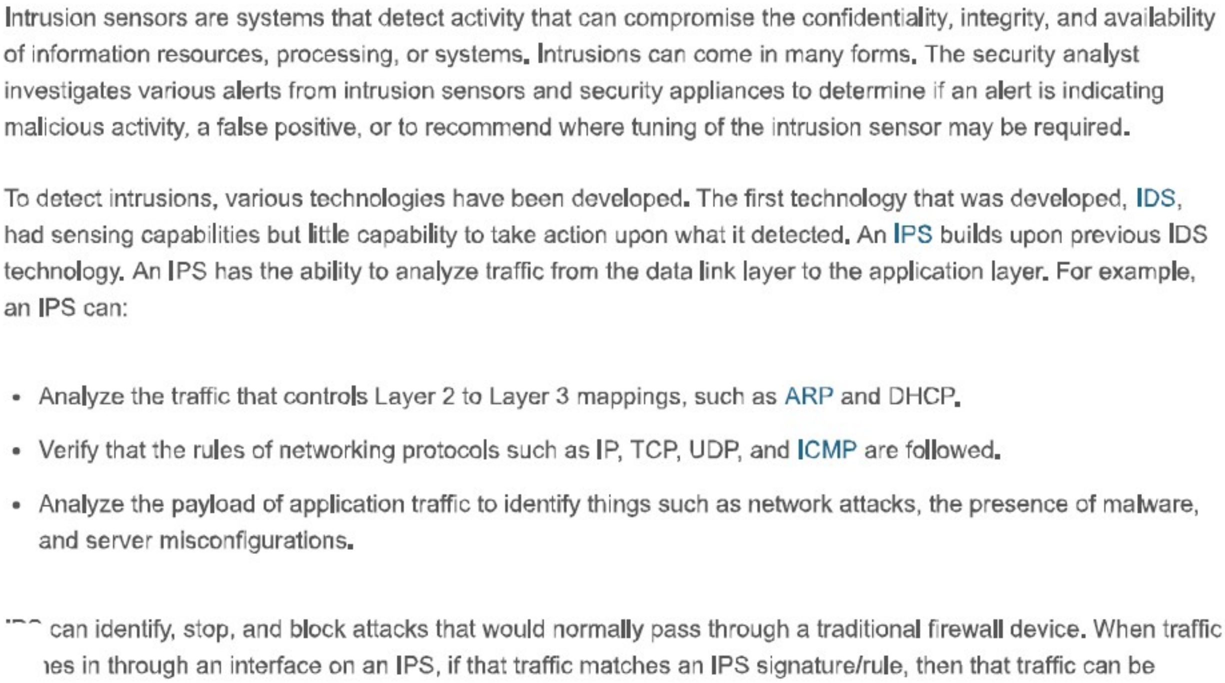
C. ICMP(ping) Flood Attack

The attack was made by flooding the victim's machine with ICMP echo packets. We use the tool hyenae. Running hyenae from command line on the attacker's system we use the following command: # hyenae -I 1 -A 4 -a icmp-echo -s %-% -d %-192.168.0.5 -t 128 Description: **-d %-192.168.0.5** is used to specify the destination ip address that is 192.168.0.5. % is used in the case where the mac address of victim machine is not known. **-s %-%** is used for spoofing the source ip address and mac address **-t 128** the time to live content of the packet which in this case is set to 128 ms. **-I 1-A 4** indicates that we are attacking from the local machine. **-a icmp-echo** indicates that we are sending icmp echo packets.


File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

```
No.    Time          Source          Destination     Protocol Length Info
  9049 19.444214000  192.168.0.6     192.168.0.5     ICMP     60 Echo (ping) request  id=0x0039, seq=6/1536, ttl=127 (reply in 9050)
  9050 19.444234000  192.168.0.5     192.168.0.6     ICMP     42 Echo (ping) reply    id=0x0039, seq=6/1536, ttl=64 (request in 9049)
  9051 19.444238000  192.168.0.6     192.168.0.5     ICMP     60 Echo (ping) request  id=0x0027, seq=111/28416, ttl=127 (reply in 9052)
  9052 19.444243000  192.168.0.5     192.168.0.6     ICMP     42 Echo (ping) reply    id=0x0027, seq=111/28416, ttl=64 (request in 9051)
  9053 19.444244000  192.168.0.6     192.168.0.5     ICMP     60 Echo (ping) request  id=0x0061, seq=148/37888, ttl=127 (reply in 9054)
  9054 19.444248000  192.168.0.5     192.168.0.6     ICMP     42 Echo (ping) reply    id=0x0061, seq=148/37888, ttl=64 (request in 9053)
  9055 19.444250000  192.168.0.6     192.168.0.5     ICMP     60 Echo (ping) request  id=0x001f, seq=40/10240, ttl=127 (reply in 9056)
  9056 19.444253000  192.168.0.5     192.168.0.6     ICMP     42 Echo (ping) reply    id=0x001f, seq=40/10240, ttl=64 (request in 9055)
  9057 19.444263000  192.168.0.6     192.168.0.5     ICMP     60 Echo (ping) request  id=0x0031, seq=124/31744, ttl=127 (reply in 9058)
  9058 19.444267000  192.168.0.5     192.168.0.6     ICMP     42 Echo (ping) reply    id=0x0031, seq=124/31744, ttl=64 (request in 9057)
  9059 19.444300000  192.168.0.6     192.168.0.5     ICMP     60 Echo (ping) request  id=0x0064, seq=97/24832, ttl=127 (reply in 9060)
  9060 19.444308000  192.168.0.5     192.168.0.6     ICMP     42 Echo (ping) reply    id=0x0064, seq=97/24832, ttl=64 (request in 9059)
  9061 19.444311000  192.168.0.6     192.168.0.5     ICMP     60 Echo (ping) request  id=0x00a2, seq=9/2304, ttl=127 (reply in 9062)
  9062 19.444316000  192.168.0.5     192.168.0.6     ICMP     42 Echo (ping) reply    id=0x00a2, seq=9/2304, ttl=64 (request in 9061)
  9063 19.445381000  192.168.0.6     192.168.0.5     ICMP     60 Echo (ping) request  id=0x006f, seq=162/41472, ttl=127 (reply in 9064)
  9064 19.445400000  192.168.0.5     192.168.0.6     ICMP     42 Echo (ping) reply    id=0x006f, seq=162/41472, ttl=64 (request in 9063)
  9065 19.445404000  192.168.0.6     192.168.0.5     ICMP     60 Echo (ping) request  id=0x003a, seq=185/47360, ttl=127 (reply in 9066)
  9066 19.445408000  192.168.0.5     192.168.0.6     ICMP     42 Echo (ping) reply    id=0x003a, seq=185/47360, ttl=64 (request in 9065)
  9067 19.445411000  192.168.0.6     192.168.0.5     ICMP     60 Echo (ping) request  id=0x0040, seq=84/21504, ttl=127 (reply in 9068)
  9068 19.445416000  192.168.0.5     192.168.0.6     ICMP     42 Echo (ping) reply    id=0x0040, seq=84/21504, ttl=64 (request in 9067)

▸ Frame 2950: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▸ Ethernet II, Src: Sony_68:1a:dc (3c:07:71:68:1a:dc), Dst: LiteonTe_43:ec:0b (b8:86:87:43:ec:0b)
▸ Internet Protocol Version 4, Src: 192.168.0.5 (192.168.0.5), Dst: 192.168.0.6 (192.168.0.6)
▸ Internet Control Message Protocol

0000  b8 86 87 45 ec 0b 3c 07  71 68 1a dc 08 00 45 00   ...E..<. qh....E.
0010  00 1c ec 66 00 00 40 01  0d 1f c0 a8 00 05 c0 a8   ...f..@. ........
0020  00 06 00 00 ff ad 00 33  00 1f                     .......3 ..

enp3s0: <live capture in progre...   Packets: 9068 · Displayed: 9068 (100.0%)   Profile: Default
```

Fig 20. ICMP echo packets received on victim machine

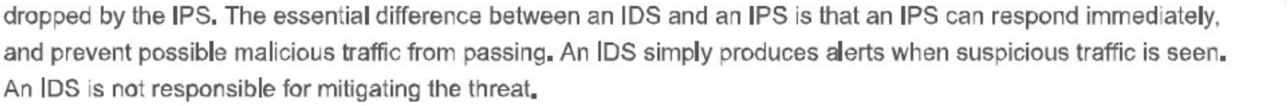


Intrusion sensors are systems that detect activity that can compromise the confidentiality, integrity, and availability of information resources, processing, or systems. Intrusions can come in many forms. The security analyst investigates various alerts from intrusion sensors and security appliances to determine if an alert is indicating malicious activity, a false positive, or to recommend where tuning of the intrusion sensor may be required.

To detect intrusions, various technologies have been developed. The first technology that was developed, IDS, had sensing capabilities but little capability to take action upon what it detected. An IPS builds upon previous IDS technology. An IPS has the ability to analyze traffic from the data link layer to the application layer. For example, an IPS can:

- Analyze the traffic that controls Layer 2 to Layer 3 mappings, such as ARP and DHCP.
- Verify that the rules of networking protocols such as IP, TCP, UDP, and ICMP are followed.
- Analyze the payload of application traffic to identify things such as network attacks, the presence of malware, and server misconfigurations.

IPS can identify, stop, and block attacks that would normally pass through a traditional firewall device. When traffic comes in through an interface on an IPS, if that traffic matches an IPS signature/rule, then that traffic can be



dropped by the IPS. The essential difference between an IDS and an IPS is that an IPS can respond immediately, and prevent possible malicious traffic from passing. An IDS simply produces alerts when suspicious traffic is seen. An IDS is not responsible for mitigating the threat.

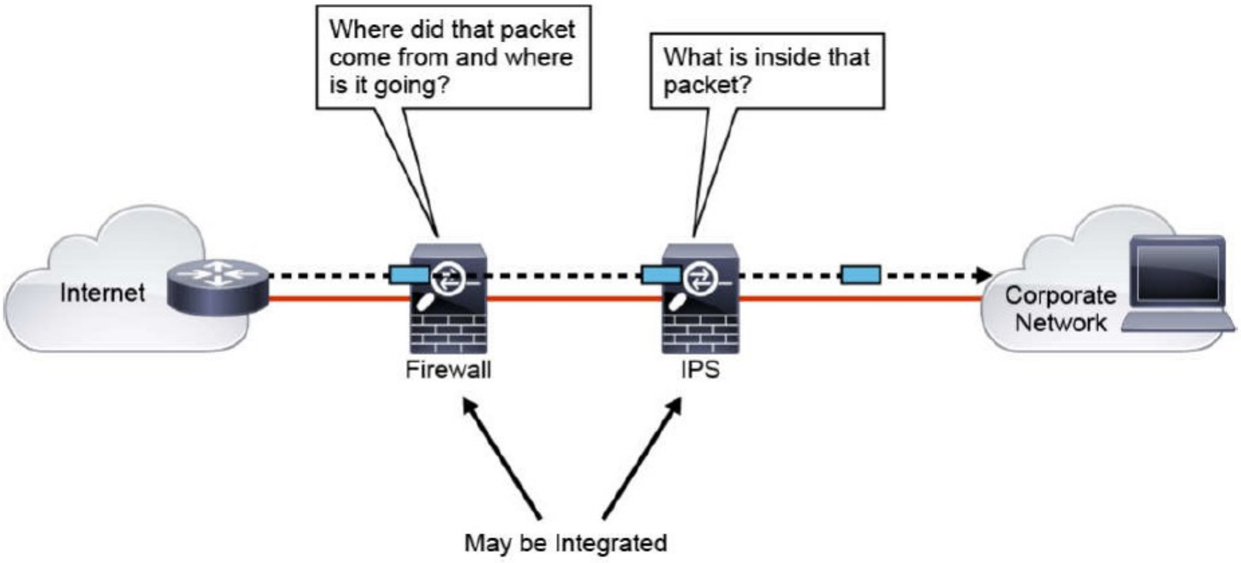
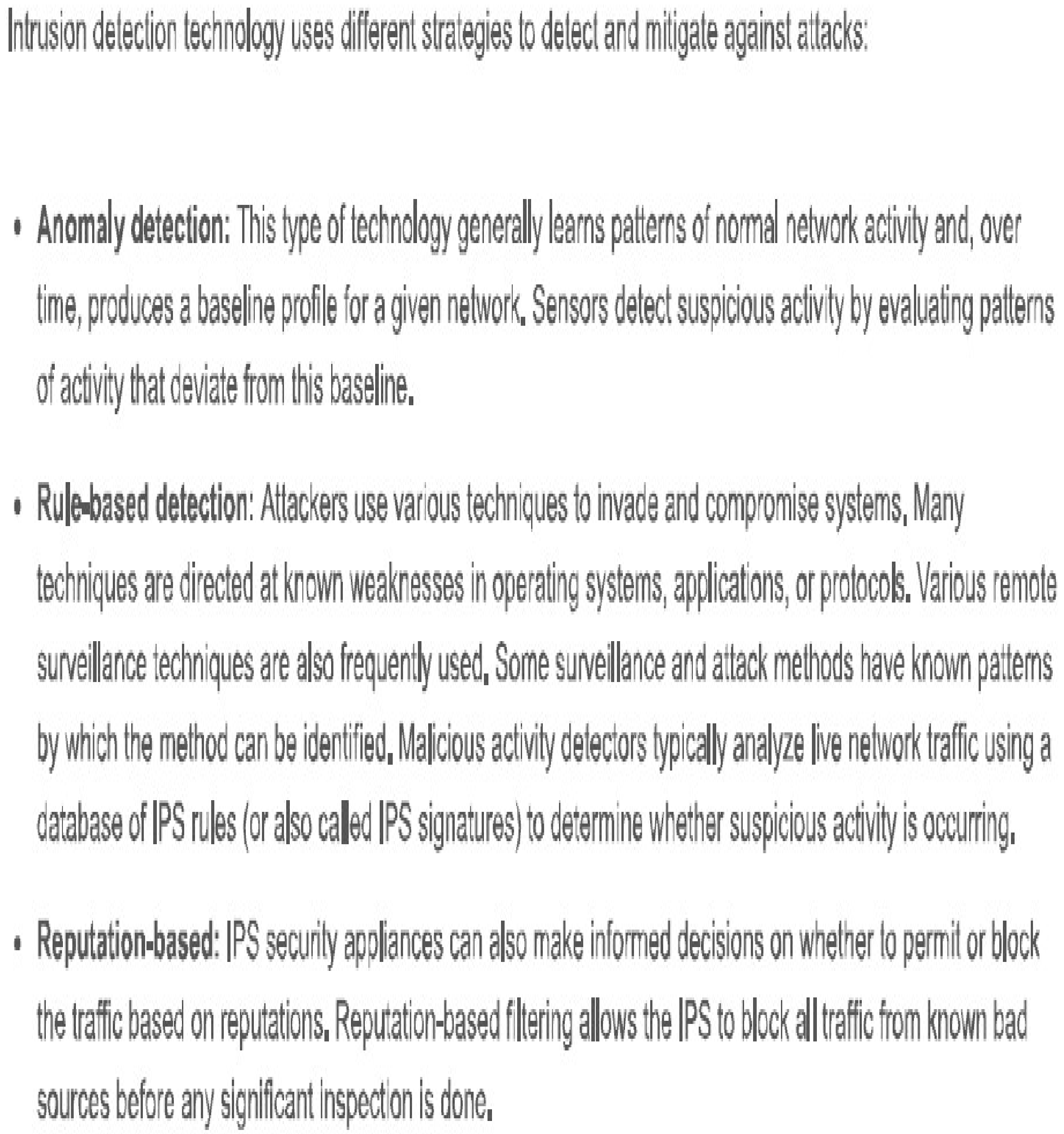

Where did that packet come from and where is it going?
What is inside that packet?
Internet
Firewall
IPS
Corporate Network
May be Integrated

Fig 21. Working in IDS



Intrusion detection technology uses different strategies to detect and mitigate against attacks:

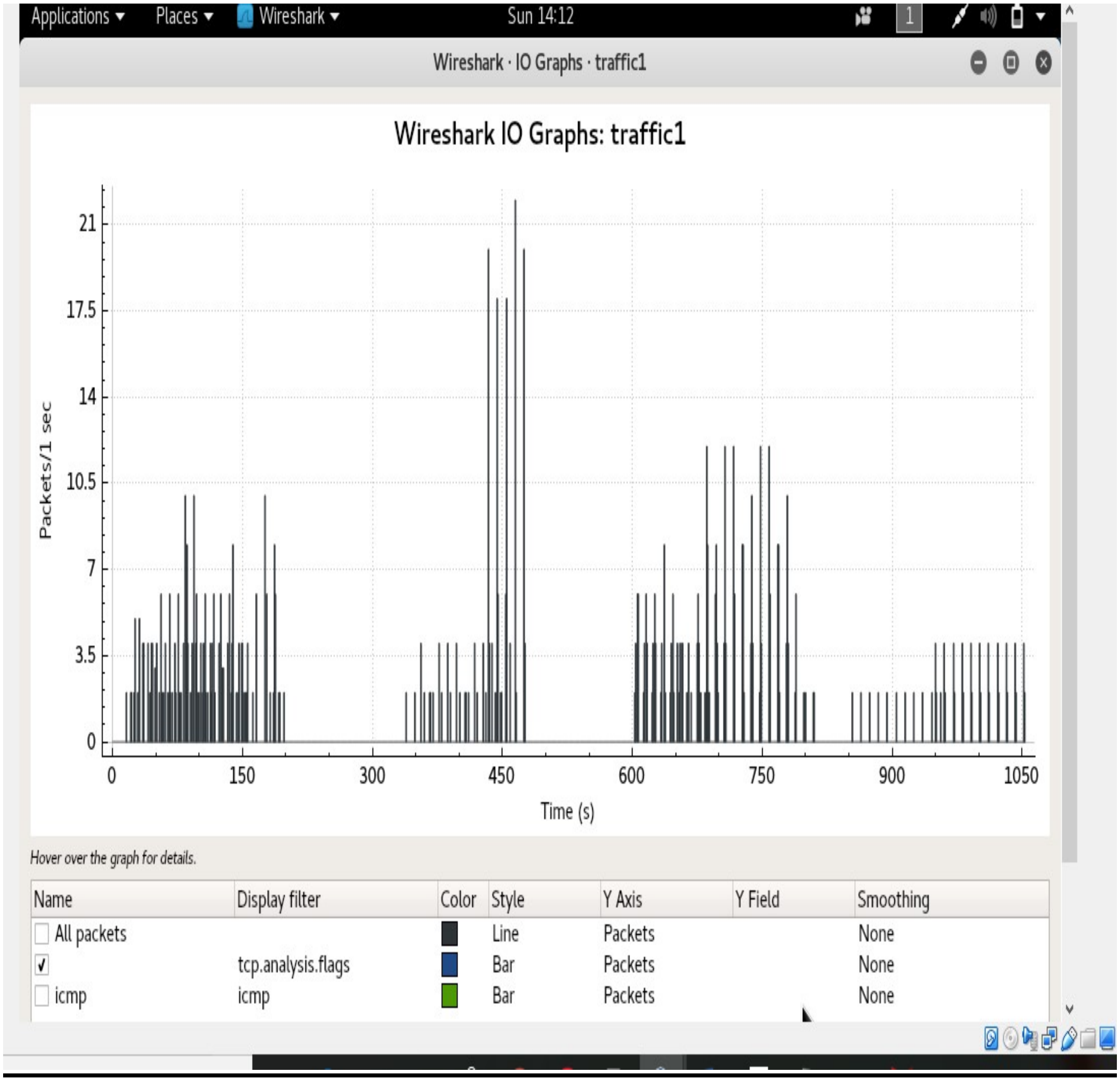
- **Anomaly detection:** This type of technology generally learns patterns of normal network activity and, over time, produces a baseline profile for a given network. Sensors detect suspicious activity by evaluating patterns of activity that deviate from this baseline.

- **Rule-based detection**: Attackers use various techniques to invade and compromise systems, Many techniques are directed at known weaknesses in operating systems, applications, or protocols. Various remote surveillance techniques are also frequently used, Some surveillance and attack methods have known patterns by which the method can be identified. Malicious activity detectors typically analyze live network traffic using a database of IPS rules (or also called IPS signatures) to determine whether suspicious activity is occurring.

- **Reputation-based:** IPS security appliances can also make informed decisions on whether to permit or block the traffic based on reputations. Reputation-based filtering allows the IPS to block all traffic from known bad sources before any significant inspection is done.

6.4 Generation of IO graph using Wireshark

IO Graphs look at all the traffic in the trace file regardless of direction whereas some other graphs (such as Round Trip Time graphs and Throughput graphs) look at traffic flowing in one way only).



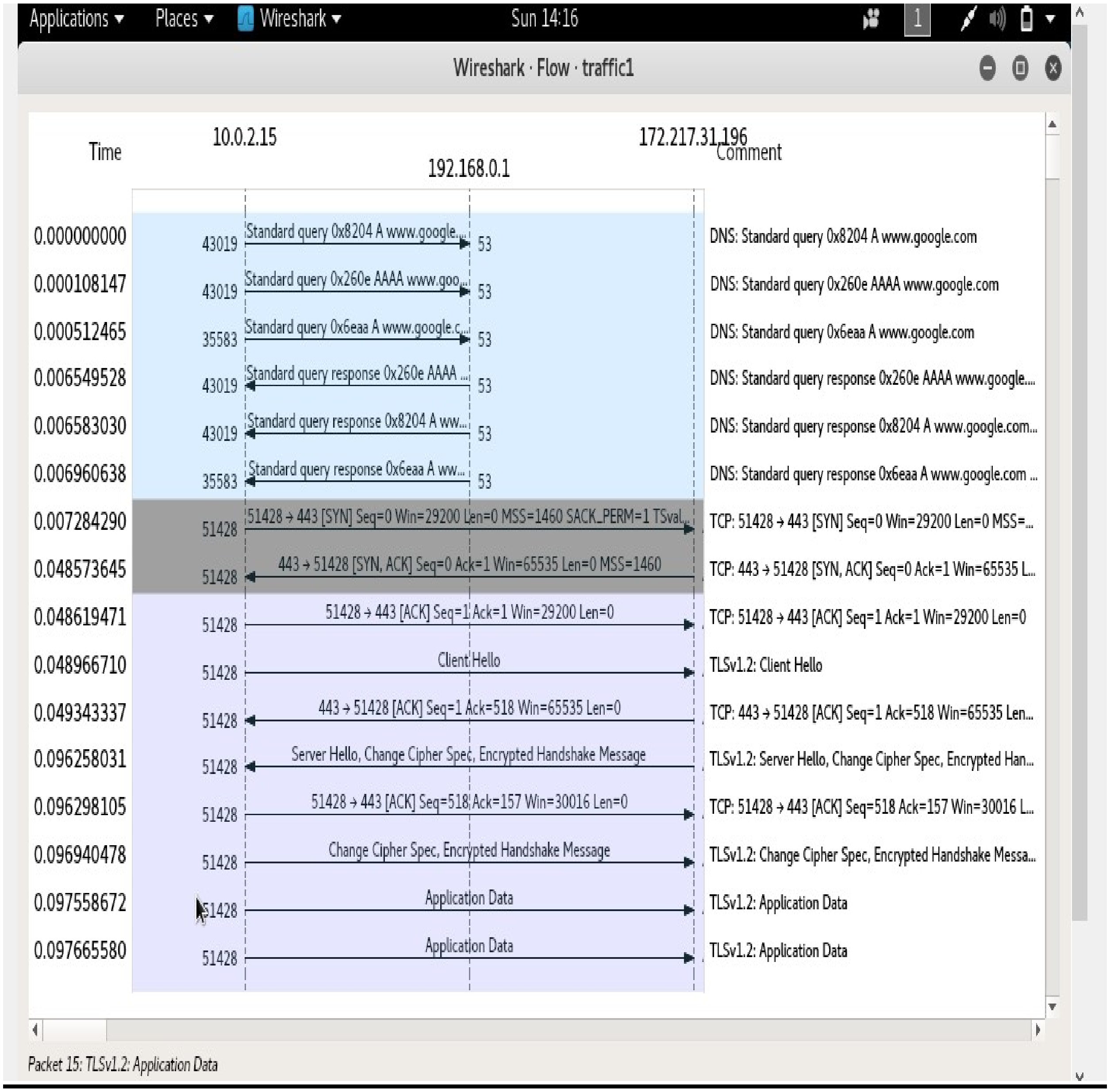
Hover over the graph for details.

Name	Display filter	Color	Style	Y Axis	Y Field	Smoothing
☐ All packets		⬛	Line	Packets		None
☑	tcp.analysis.flags	🟦	Bar	Packets		None
☐ icmp	icmp	🟩	Bar	Packets		None

Fig 22. Generation of IO graph

6.5 Generation of Flow graph using Wireshark

Flow graphs create a packet-by-packet interpretation of the traffic, separating source and target hosts by columns. This is particularly useful when interpreting HTTP,TCP,ICMP,UDP traffic.

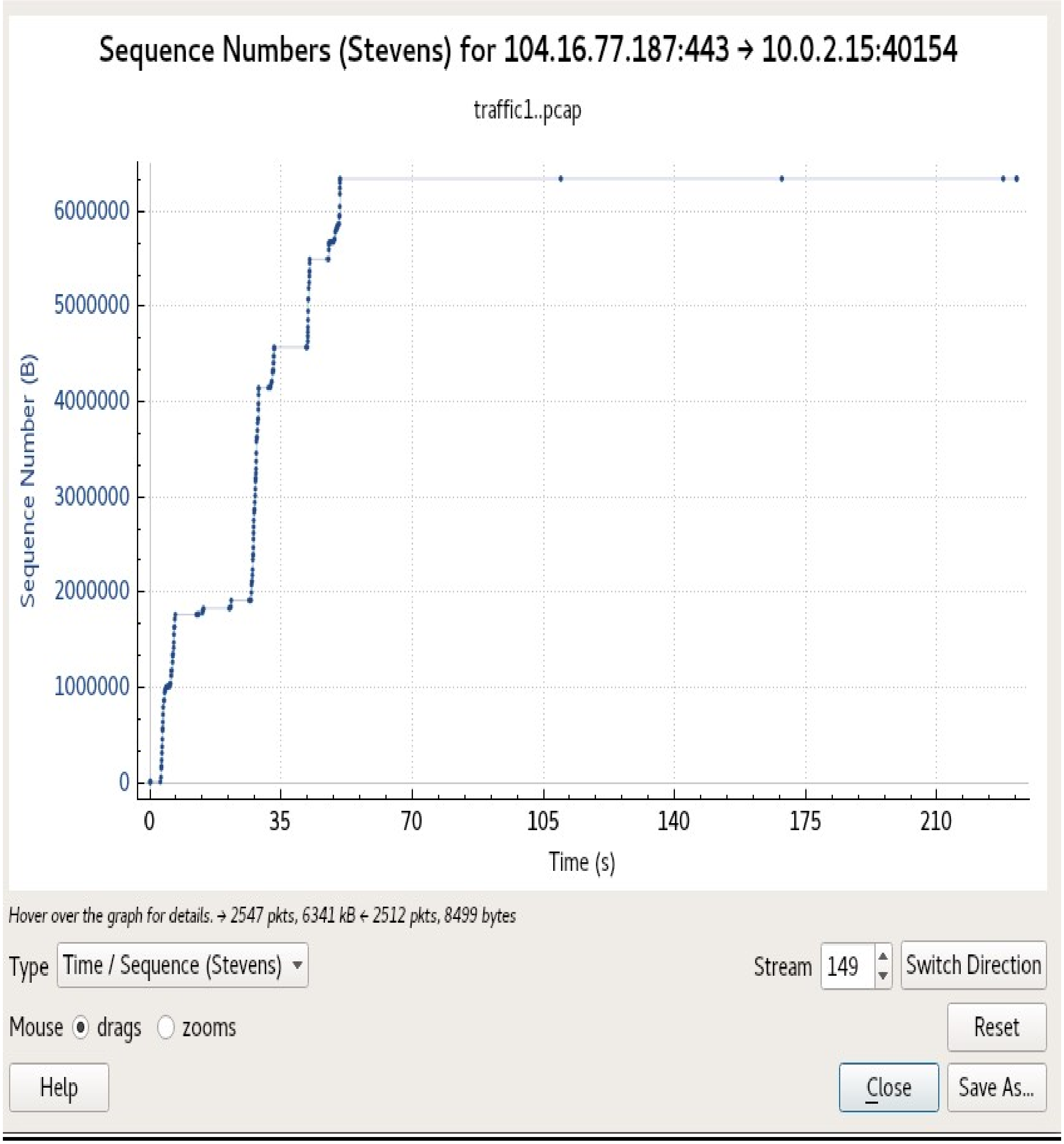


Time	10.0.2.15		192.168.0.1	172.217.31.196	Comment
0.000000000	43019	Standard query 0x8204 A www.google...	53		DNS: Standard query 0x8204 A www.google.com
0.000108147	43019	Standard query 0x260e AAAA www.goo...	53		DNS: Standard query 0x260e AAAA www.google.com
0.000512465	35583	Standard query 0x6eaa A www.google.c...	53		DNS: Standard query 0x6eaa A www.google.com
0.006549528	43019	Standard query response 0x260e AAAA ...	53		DNS: Standard query response 0x260e AAAA www.google....
0.006583030	43019	Standard query response 0x8204 A ww...	53		DNS: Standard query response 0x8204 A www.google.com...
0.006960638	35583	Standard query response 0x6eaa A ww...	53		DNS: Standard query response 0x6eaa A www.google.com ...
0.007284290	51428	51428 → 443 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval...			TCP: 51428 → 443 [SYN] Seq=0 Win=29200 Len=0 MSS=...
0.048573645	51428	443 → 51428 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460			TCP: 443 → 51428 [SYN, ACK] Seq=0 Ack=1 Win=65535 L...
0.048619471	51428	51428 → 443 [ACK] Seq=1 Ack=1 Win=29200 Len=0			TCP: 51428 → 443 [ACK] Seq=1 Ack=1 Win=29200 Len=0
0.048966710	51428	Client Hello			TLSv1.2: Client Hello
0.049343337	51428	443 → 51428 [ACK] Seq=1 Ack=518 Win=65535 Len=0			TCP: 443 → 51428 [ACK] Seq=1 Ack=518 Win=65535 Len...
0.096258031	51428	Server Hello, Change Cipher Spec, Encrypted Handshake Message			TLSv1.2: Server Hello, Change Cipher Spec, Encrypted Han...
0.096298105	51428	51428 → 443 [ACK] Seq=518 Ack=157 Win=30016 Len=0			TCP: 51428 → 443 [ACK] Seq=518 Ack=157 Win=30016 L...
0.096940478	51428	Change Cipher Spec, Encrypted Handshake Message			TLSv1.2: Change Cipher Spec, Encrypted Handshake Messa...
0.097558672	51428	Application Data			TLSv1.2: Application Data
0.097665580	51428	Application Data			TLSv1.2: Application Data

Packet 15: TLSv1.2: Application Data

Fig 23. Generation of flow graph

6.6 Generation of TCP Stream graph using Wireshark



Hover over the graph for details. → 2547 pkts, 6341 kB ← 2512 pkts, 8499 bytes

Type Time / Sequence (Stevens)

Stream 149

Mouse ● drags ○ zooms

Fig 24. Generating TCP Stream graph

6.7 Protocol Hierarchy using Wireshark

This is a tree of all the protocols in the capture. Each row contains the statistical values of one protocol. Two of the columns (*Percent Packets* and *Percent Bytes*) serve double duty as bar graphs. If a display filter is set it will be shown at the bottom.

Protocol Hierarchy Columns

Protocol
This protocol's name

Percent Packets
The percentage of protocol packets relative to all packets in the capture

Packets
The total number of packets of this protocol

Percent Bytes
The percentage of protocol bytes relative to the total bytes in the capture

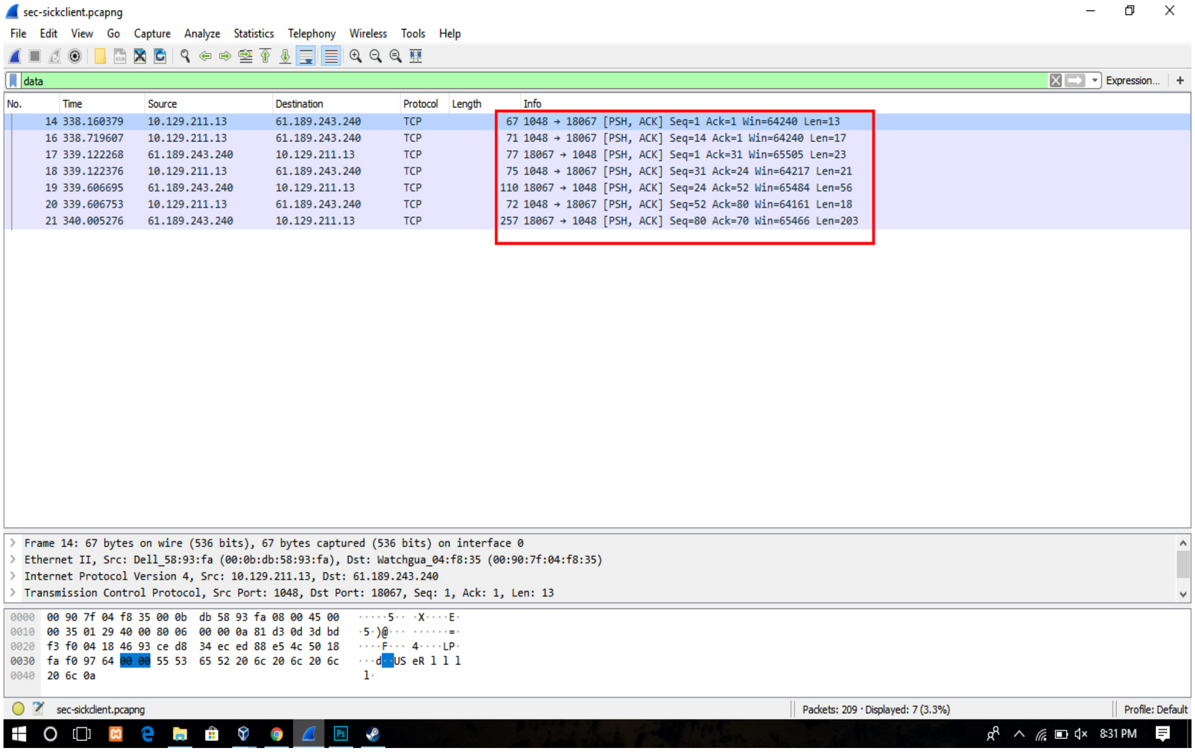
Bytes
The total number of bytes of this protocol

Bits/s
The bandwidth of this protocol relative to the capture time

End Packets
The absolute number of packets of this protocol where it was the highest protocol in the stack (last dissected)

End Bytes
The absolute number of bytes of this protocol where it was the highest protocol in the stack (last dissected)

End Bits/s
The bandwidth of this protocol relative to the capture time where was the highest protocol in the stack (last dissected)

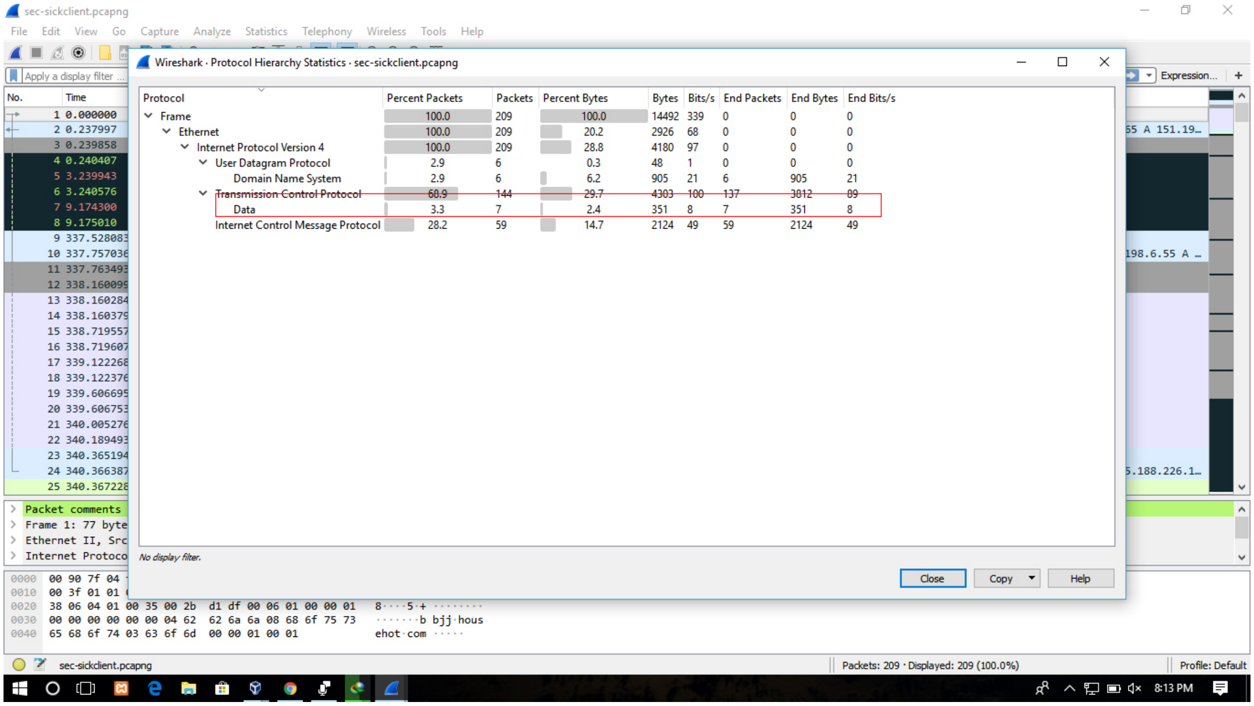

sec-sickclient.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
data
Expression…
Packets: 209 · Displayed: 7 (3.3%)
Profile: Default
8:31 PM

```
No.   Time        Source          Destination     Protocol Length Info
   14 338.160379  10.129.211.13   61.189.243.240  TCP      67 1048 → 18067 [PSH, ACK] Seq=1 Ack=1 Win=64240 Len=13
   16 338.719607  10.129.211.13   61.189.243.240  TCP      71 1048 → 18067 [PSH, ACK] Seq=14 Ack=1 Win=64240 Len=17
   17 339.122268  61.189.243.240  10.129.211.13   TCP      77 18067 → 1048 [PSH, ACK] Seq=1 Ack=31 Win=65505 Len=23
   18 339.122376  10.129.211.13   61.189.243.240  TCP      75 1048 → 18067 [PSH, ACK] Seq=31 Ack=24 Win=64217 Len=21
   19 339.606695  61.189.243.240  10.129.211.13   TCP     110 18067 → 1048 [PSH, ACK] Seq=24 Ack=52 Win=65484 Len=56
   20 339.606753  10.129.211.13   61.189.243.240  TCP      72 1048 → 18067 [PSH, ACK] Seq=52 Ack=80 Win=64161 Len=18
   21 340.005276  61.189.243.240  10.129.211.13   TCP     257 18067 → 1048 [PSH, ACK] Seq=80 Ack=70 Win=65466 Len=203
```

```
> Frame 14: 67 bytes on wire (536 bits), 67 bytes captured (536 bits) on interface 0
> Ethernet II, Src: Dell_58:93:fa (00:0b:db:58:93:fa), Dst: Watchgua_04:f8:35 (00:90:7f:04:f8:35)
> Internet Protocol Version 4, Src: 10.129.211.13, Dst: 61.189.243.240
> Transmission Control Protocol, Src Port: 1048, Dst Port: 18067, Seq: 1, Ack: 1, Len: 13
```

```
0000  00 90 7f 04 f8 35 00 0b  db 58 93 fa 08 00 45 00   ·····5·· ·X····E·
0010  00 35 01 29 40 00 80 06  00 00 0a 81 d3 0d 3d bd   ·5·)@··· ······=·
0020  f3 f0 04 18 46 93 ce d8  34 ec ed 88 e5 4c 50 18   ····F··· 4····LP·
0030  fa f0 97 64 00 00 55 53  65 52 20 6c 20 6c 20 6c   ···d··US eR l l l
0040  20 6c 0a                                           l·
```

Fig 25. Protocol hierarchy of a data path


sec-sickclient.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter …
Wireshark · Protocol Hierarchy Statistics · sec-sickclient.pcapng
No display filter.
Close
Copy
Help
Packets: 209 · Displayed: 209 (100.0%)
Profile: Default
8:13 PM

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	209	100.0	14492	339	0	0	0
Ethernet	100.0	209	20.2	2926	68	0	0	0
Internet Protocol Version 4	100.0	209	28.8	4180	97	0	0	0
User Datagram Protocol	2.9	6	0.3	48	1	0	0	0
Domain Name System	2.9	6	6.2	905	21	6	905	21
Transmission Control Protocol	68.9	144	29.7	4303	100	137	3812	89
Data	3.3	7	2.4	351	8	7	351	8
Internet Control Message Protocol	28.2	59	14.7	2124	49	59	2124	49

Fig 26. Protocol hierarchy window in Wireshark

Detecting whether the packet is malicious or not using protocol hierarchy

So, from the above picture given we can determine that all the packets are IPV4 packets. Then we have some UDP packets which is basically the DNS, then directly under the TCP packets the word DATA is shown which means that Wireshark does not recognize the application that are running over TCP which it dropped as DATA. After selecting the DATA and applying filter over it. It will automatically create a hierarchy called Filter. Now when we look into the data packets Wireshark does not recognize the destination port no. and the traffic is IRC TRAFFIC travelling over a non-standard port no. Thus, detected to be a malicious packet.

6.8 Configuring snort to detect pings

Snort starts with a long set of default configuration, but we will start with a very simple ping detector.

In a terminal window, enter the commands, pressing enter to run the commands.

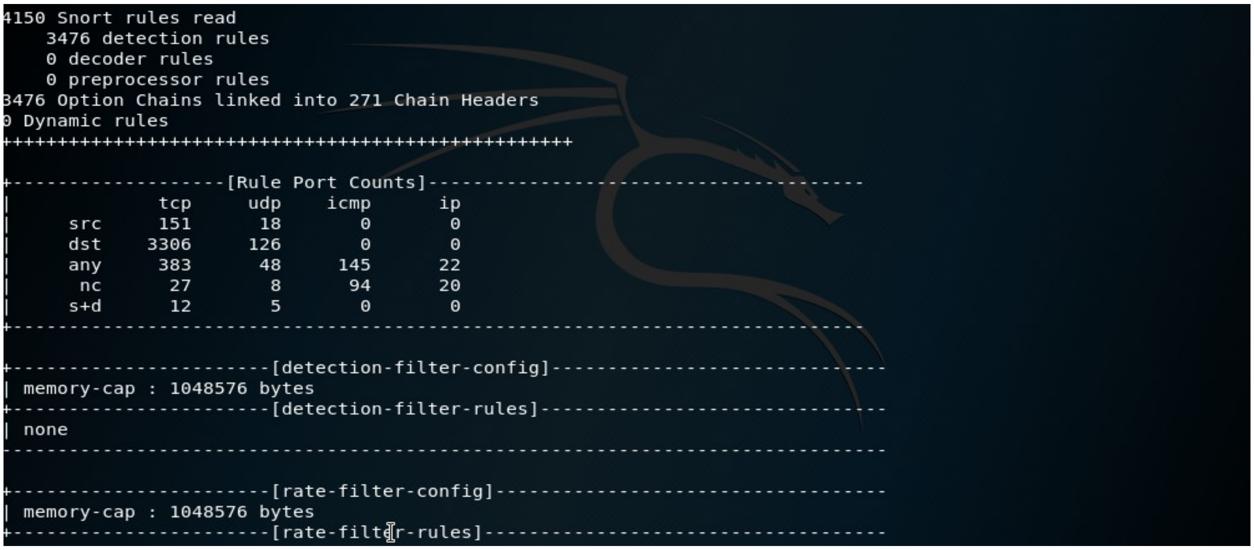
Cd /etc/snort

Snort-test.conf

include /etc/snort/icmp-test.rules

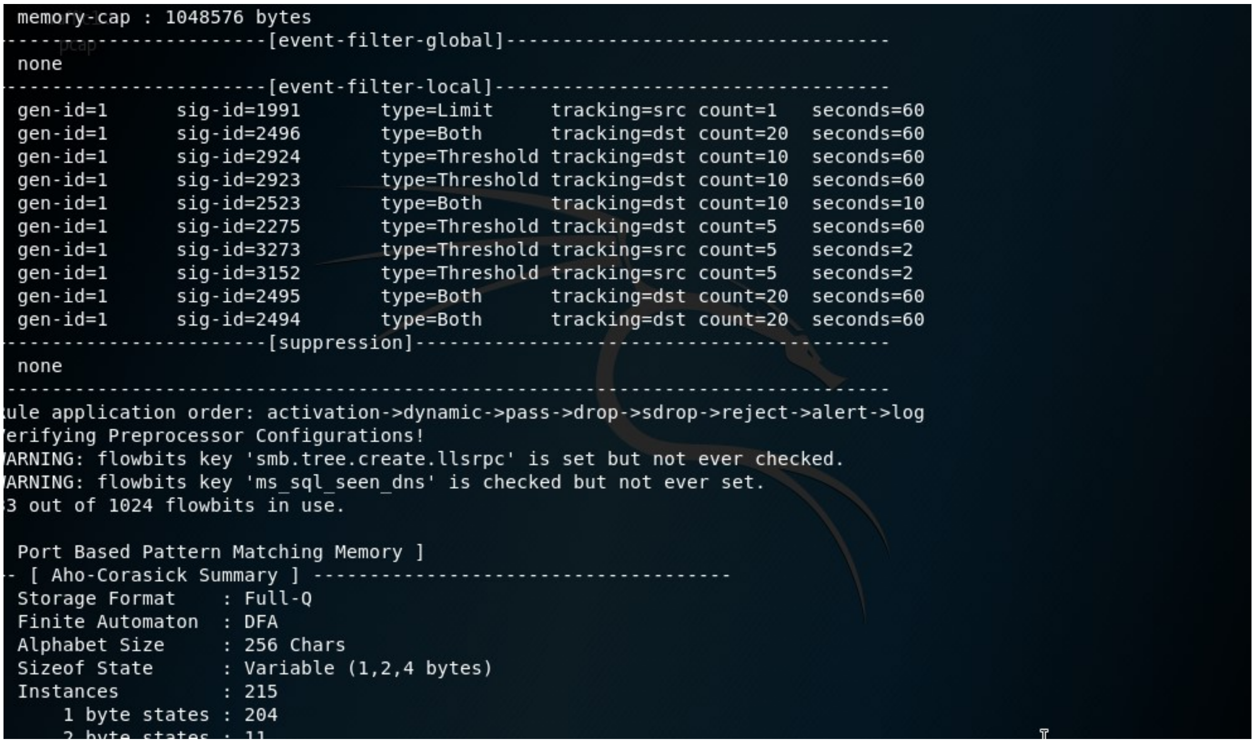
include /etc/snort/tcp-

test.rules



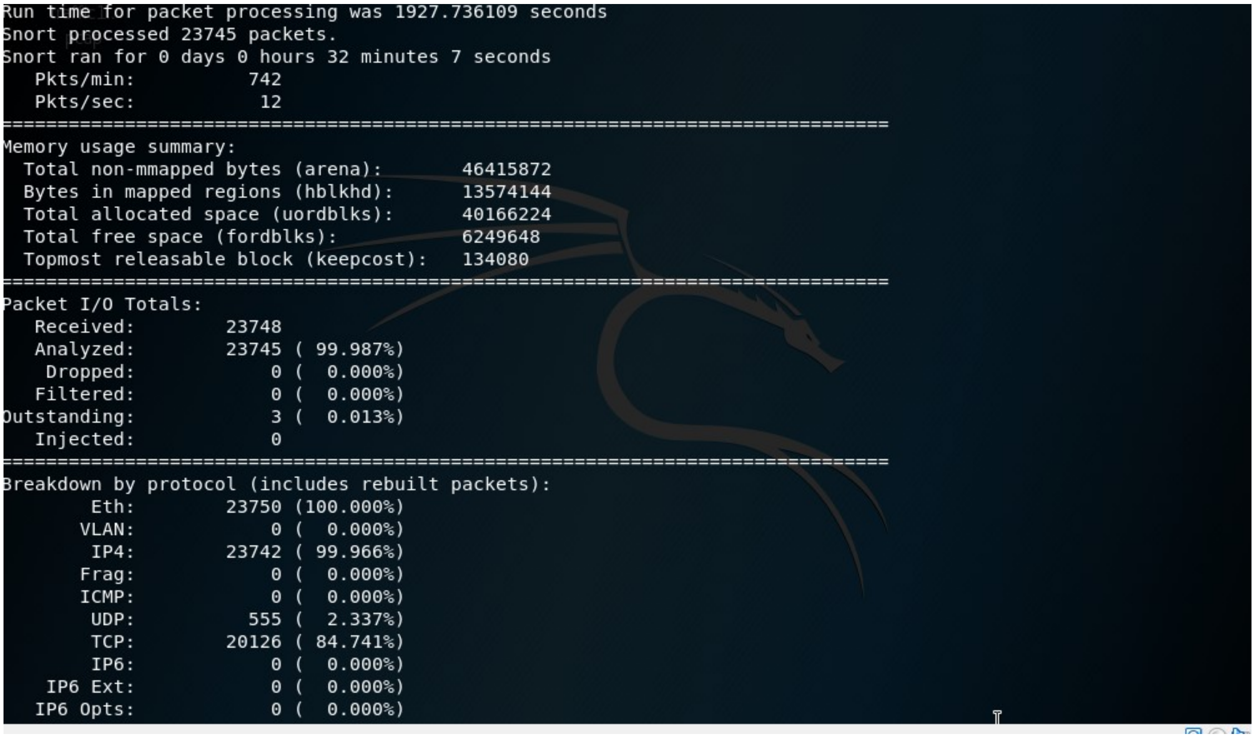
```
4150 Snort rules read
    3476 detection rules
    0 decoder rules
    0 preprocessor rules
3476 Option Chains linked into 271 Chain Headers
0 Dynamic rules
+++++++++++++++++++++++++++++++++++++++++++++++++++

+-------------------[Rule Port Counts]---------------------------------------
|             tcp     udp    icmp      ip
|     src     151      18       0       0
|     dst    3306     126       0       0
|     any     383      48     145      22
|      nc      27       8      94      20
|     s+d      12       5       0       0
+----------------------------------------------------------------------------

+-----------------------[detection-filter-config]------------------------------
| memory-cap : 1048576 bytes
+-----------------------[detection-filter-rules]-------------------------------
| none
-------------------------------------------------------------------------------

+-----------------------[rate-filter-config]-----------------------------------
| memory-cap : 1048576 bytes
+-----------------------[rate-filter-rules]------------------------------------
```

Fig 27. Detection of ICMP packets using Snort



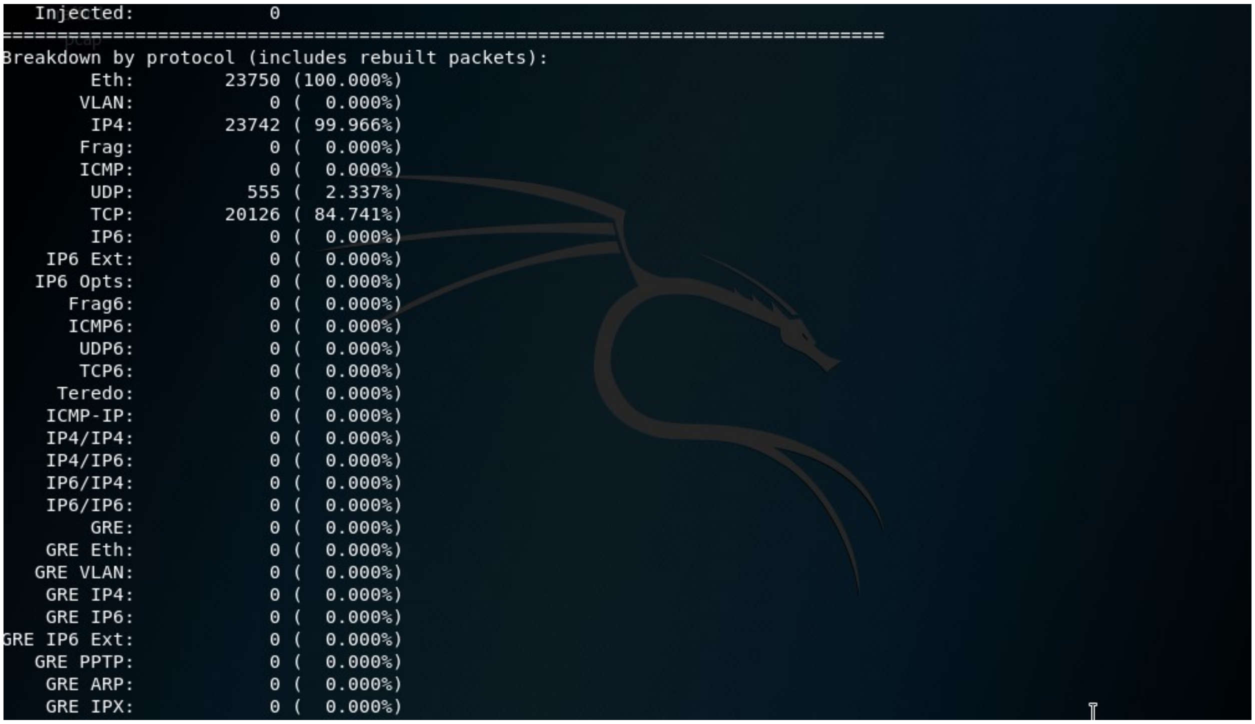
```
 memory-cap : 1048576 bytes
-----------------------[event-filter-global]----------------------------------
 none
-----------------------[event-filter-local]-----------------------------------
 gen-id=1      sig-id=1991       type=Limit     tracking=src count=1   seconds=60
 gen-id=1      sig-id=2496       type=Both      tracking=dst count=20  seconds=60
 gen-id=1      sig-id=2924       type=Threshold tracking=dst count=10  seconds=60
 gen-id=1      sig-id=2923       type=Threshold tracking=dst count=10  seconds=60
 gen-id=1      sig-id=2523       type=Both      tracking=dst count=10  seconds=10
 gen-id=1      sig-id=2275       type=Threshold tracking=dst count=5   seconds=60
 gen-id=1      sig-id=3273       type=Threshold tracking=src count=5   seconds=2
 gen-id=1      sig-id=3152       type=Threshold tracking=src count=5   seconds=2
 gen-id=1      sig-id=2495       type=Both      tracking=dst count=20  seconds=60
 gen-id=1      sig-id=2494       type=Both      tracking=dst count=20  seconds=60
-----------------------[suppression]------------------------------------------
 none
------------------------------------------------------------------------------
ule application order: activation->dynamic->pass->drop->sdrop->reject->alert->log
erifying Preprocessor Configurations!
ARNING: flowbits key 'smb.tree.create.llsrpc' is set but not ever checked.
ARNING: flowbits key 'ms_sql_seen_dns' is checked but not ever set.
3 out of 1024 flowbits in use.

 Port Based Pattern Matching Memory ]
- [ Aho-Corasick Summary ] -------------------------------------
 Storage Format    : Full-Q
 Finite Automaton  : DFA
 Alphabet Size     : 256 Chars
 Sizeof State      : Variable (1,2,4 bytes)
 Instances         : 215
     1 byte states : 204
     2 byte states : 11
```

Fig 28. Detection of packets using Snort



```
Run time for packet processing was 1927.736109 seconds
Snort processed 23745 packets.
Snort ran for 0 days 0 hours 32 minutes 7 seconds
   Pkts/min:          742
   Pkts/sec:           12
===============================================================================
Memory usage summary:
  Total non-mmapped bytes (arena):       46415872
  Bytes in mapped regions (hblkhd):      13574144
  Total allocated space (uordblks):      40166224
  Total free space (fordblks):           6249648
  Topmost releasable block (keepcost):   134080
===============================================================================
Packet I/O Totals:
   Received:        23748
   Analyzed:        23745 ( 99.987%)
    Dropped:            0 (  0.000%)
   Filtered:            0 (  0.000%)
Outstanding:            3 (  0.013%)
   Injected:            0
===============================================================================
Breakdown by protocol (includes rebuilt packets):
        Eth:        23750 (100.000%)
       VLAN:            0 (  0.000%)
        IP4:        23742 ( 99.966%)
       Frag:            0 (  0.000%)
       ICMP:            0 (  0.000%)
        UDP:          555 (  2.337%)
        TCP:        20126 ( 84.741%)
        IP6:            0 (  0.000%)
    IP6 Ext:            0 (  0.000%)
   IP6 Opts:            0 (  0.000%)
```

Fig 29. Detection of TCP packets using Snort



```
  Injected:            0
================================================================================
Breakdown by protocol (includes rebuilt packets):
        Eth:        23750 (100.000%)
       VLAN:            0 (  0.000%)
        IP4:        23742 ( 99.966%)
       Frag:            0 (  0.000%)
       ICMP:            0 (  0.000%)
        UDP:          555 (  2.337%)
        TCP:        20126 ( 84.741%)
        IP6:            0 (  0.000%)
    IP6 Ext:            0 (  0.000%)
   IP6 Opts:            0 (  0.000%)
      Frag6:            0 (  0.000%)
      ICMP6:            0 (  0.000%)
       UDP6:            0 (  0.000%)
       TCP6:            0 (  0.000%)
     Teredo:            0 (  0.000%)
    ICMP-IP:            0 (  0.000%)
    IP4/IP4:            0 (  0.000%)
    IP4/IP6:            0 (  0.000%)
    IP6/IP4:            0 (  0.000%)
    IP6/IP6:            0 (  0.000%)
        GRE:            0 (  0.000%)
    GRE Eth:            0 (  0.000%)
   GRE VLAN:            0 (  0.000%)
    GRE IP4:            0 (  0.000%)
    GRE IP6:            0 (  0.000%)
GRE IP6 Ext:            0 (  0.000%)
   GRE PPTP:            0 (  0.000%)
    GRE ARP:            0 (  0.000%)
    GRE IPX:            0 (  0.000%)
```

Fig 30. Breakdown of TCP & UDP Packets

6.9 Generation of alert files using Snort intrusion detection system

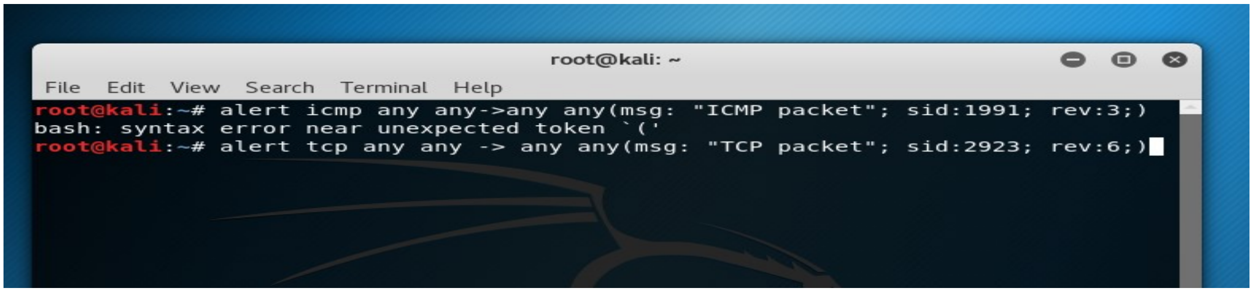
In a terminal execute this command,

Nano icmp-test.rules

Nano tcp-test rules

alert icmp any any -> any any (msg "ICMP packet"; sid: 1991; rev :3;)

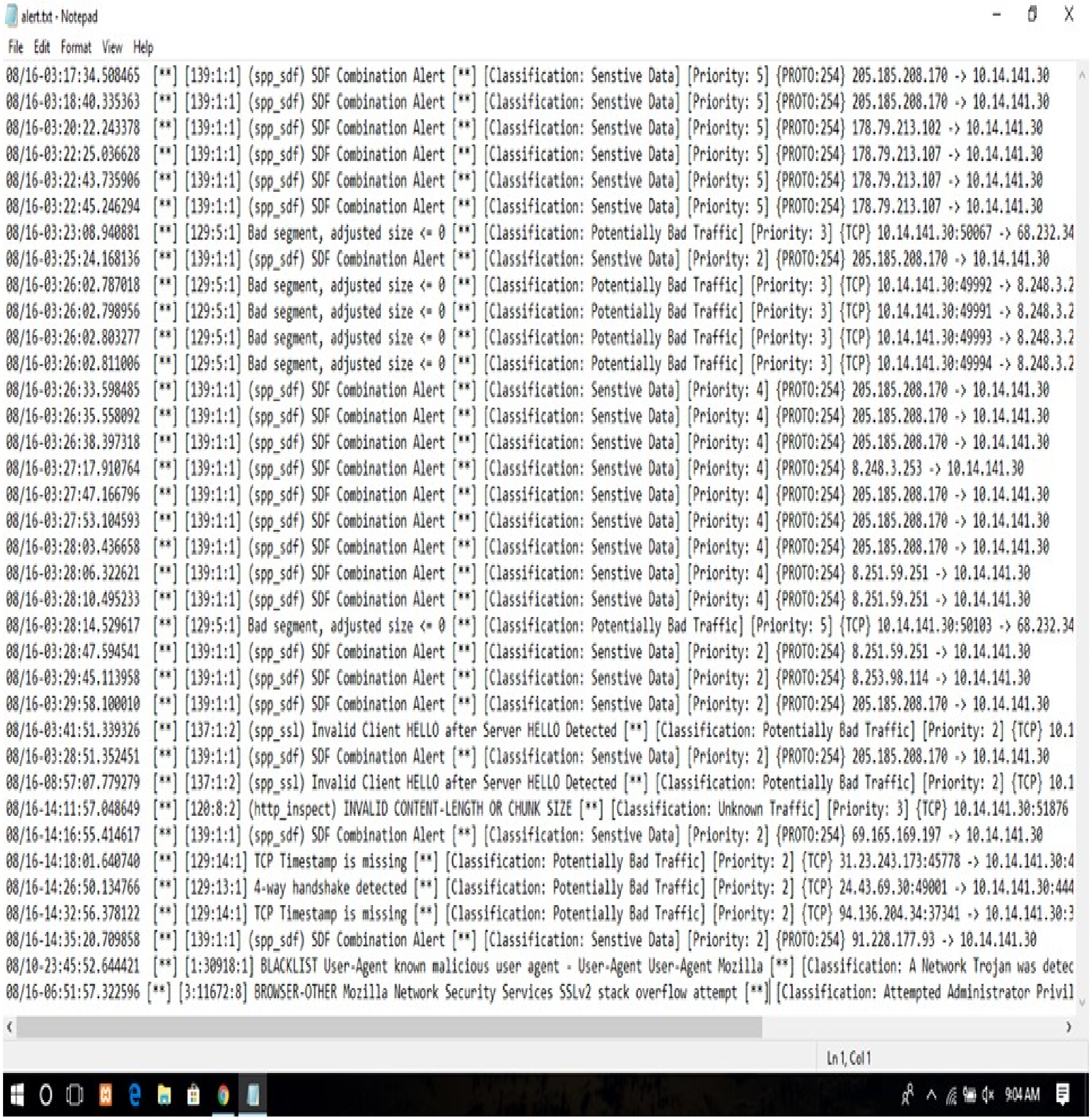
alert tcp any any -> any any (msg "TCP packet"; sid: 2923; rev :6;)



```
root@kali:~# alert icmp any any->any any(msg: "ICMP packet"; sid:1991; rev:3;)
bash: syntax error near unexpected token `('
root@kali:~# alert tcp any any -> any any(msg: "TCP packet"; sid:2923; rev:6;)
```

Fig 31. Generation of alerts in terminal

After generating the alerts , we will get an alert log file and the Pcap file which will be used for the creation of attack graph and the priority graph. As, the alerts will generate a log file which concludes the priority of the packets in which if any packet has the higher priority. They have the tendency to flood in the network.



```
08/16-03:17:34.508465  [**] [139:1:1] (spp_sdf) SDF Combination Alert [**] [Classification: Senstive Data] [Priority: 5] {PROTO:254} 205.185.208.170 -> 10.14.141.30
08/16-03:18:40.335363  [**] [139:1:1] (spp_sdf) SDF Combination Alert [**] [Classification: Senstive Data] [Priority: 5] {PROTO:254} 205.185.208.170 -> 10.14.141.30
08/16-03:20:22.243378  [**] [139:1:1] (spp_sdf) SDF Combination Alert [**] [Classification: Senstive Data] [Priority: 5] {PROTO:254} 178.79.213.102 -> 10.14.141.30
08/16-03:22:25.036628  [**] [139:1:1] (spp_sdf) SDF Combination Alert [**] [Classification: Senstive Data] [Priority: 5] {PROTO:254} 178.79.213.107 -> 10.14.141.30
08/16-03:22:43.735906  [**] [139:1:1] (spp_sdf) SDF Combination Alert [**] [Classification: Senstive Data] [Priority: 5] {PROTO:254} 178.79.213.107 -> 10.14.141.30
08/16-03:22:45.246294  [**] [139:1:1] (spp_sdf) SDF Combination Alert [**] [Classification: Senstive Data] [Priority: 5] {PROTO:254} 178.79.213.107 -> 10.14.141.30
08/16-03:23:08.940881  [**] [129:5:1] Bad segment, adjusted size <= 0 [**] [Classification: Potentially Bad Traffic] [Priority: 3] {TCP} 10.14.141.30:50067 -> 68.232.34
08/16-03:25:24.168136  [**] [139:1:1] (spp_sdf) SDF Combination Alert [**] [Classification: Senstive Data] [Priority: 2] {PROTO:254} 205.185.208.170 -> 10.14.141.30
08/16-03:26:02.787018  [**] [129:5:1] Bad segment, adjusted size <= 0 [**] [Classification: Potentially Bad Traffic] [Priority: 3] {TCP} 10.14.141.30:49992 -> 8.248.3.2
08/16-03:26:02.798956  [**] [129:5:1] Bad segment, adjusted size <= 0 [**] [Classification: Potentially Bad Traffic] [Priority: 3] {TCP} 10.14.141.30:49991 -> 8.248.3.2
08/16-03:26:02.803277  [**] [129:5:1] Bad segment, adjusted size <= 0 [**] [Classification: Potentially Bad Traffic] [Priority: 3] {TCP} 10.14.141.30:49993 -> 8.248.3.2
08/16-03:26:02.811006  [**] [129:5:1] Bad segment, adjusted size <= 0 [**] [Classification: Potentially Bad Traffic] [Priority: 3] {TCP} 10.14.141.30:49994 -> 8.248.3.2
08/16-03:26:33.598485  [**] [139:1:1] (spp_sdf) SDF Combination Alert [**] [Classification: Senstive Data] [Priority: 4] {PROTO:254} 205.185.208.170 -> 10.14.141.30
08/16-03:26:35.558092  [**] [139:1:1] (spp_sdf) SDF Combination Alert [**] [Classification: Senstive Data] [Priority: 4] {PROTO:254} 205.185.208.170 -> 10.14.141.30
08/16-03:26:38.397318  [**] [139:1:1] (spp_sdf) SDF Combination Alert [**] [Classification: Senstive Data] [Priority: 4] {PROTO:254} 205.185.208.170 -> 10.14.141.30
08/16-03:27:17.910764  [**] [139:1:1] (spp_sdf) SDF Combination Alert [**] [Classification: Senstive Data] [Priority: 4] {PROTO:254} 8.248.3.253 -> 10.14.141.30
08/16-03:27:47.166796  [**] [139:1:1] (spp_sdf) SDF Combination Alert [**] [Classification: Senstive Data] [Priority: 4] {PROTO:254} 205.185.208.170 -> 10.14.141.30
08/16-03:27:53.104593  [**] [139:1:1] (spp_sdf) SDF Combination Alert [**] [Classification: Senstive Data] [Priority: 4] {PROTO:254} 205.185.208.170 -> 10.14.141.30
08/16-03:28:03.436658  [**] [139:1:1] (spp_sdf) SDF Combination Alert [**] [Classification: Senstive Data] [Priority: 4] {PROTO:254} 205.185.208.170 -> 10.14.141.30
08/16-03:28:06.322621  [**] [139:1:1] (spp_sdf) SDF Combination Alert [**] [Classification: Senstive Data] [Priority: 4] {PROTO:254} 8.251.59.251 -> 10.14.141.30
08/16-03:28:10.495233  [**] [139:1:1] (spp_sdf) SDF Combination Alert [**] [Classification: Senstive Data] [Priority: 4] {PROTO:254} 8.251.59.251 -> 10.14.141.30
08/16-03:28:14.529617  [**] [129:5:1] Bad segment, adjusted size <= 0 [**] [Classification: Potentially Bad Traffic] [Priority: 5] {TCP} 10.14.141.30:50103 -> 68.232.34
08/16-03:28:47.594541  [**] [139:1:1] (spp_sdf) SDF Combination Alert [**] [Classification: Senstive Data] [Priority: 2] {PROTO:254} 8.251.59.251 -> 10.14.141.30
08/16-03:29:45.113958  [**] [139:1:1] (spp_sdf) SDF Combination Alert [**] [Classification: Senstive Data] [Priority: 2] {PROTO:254} 8.253.98.114 -> 10.14.141.30
08/16-03:29:58.100010  [**] [139:1:1] (spp_sdf) SDF Combination Alert [**] [Classification: Senstive Data] [Priority: 2] {PROTO:254} 205.185.208.170 -> 10.14.141.30
08/16-03:41:51.339326  [**] [137:1:2] (spp_ssl) Invalid Client HELLO after Server HELLO Detected [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 10.1
08/16-03:28:51.352451  [**] [139:1:1] (spp_sdf) SDF Combination Alert [**] [Classification: Senstive Data] [Priority: 2] {PROTO:254} 205.185.208.170 -> 10.14.141.30
08/16-08:57:07.779279  [**] [137:1:2] (spp_ssl) Invalid Client HELLO after Server HELLO Detected [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 10.1
08/16-14:11:57.048649  [**] [120:8:2] (http_inspect) INVALID CONTENT-LENGTH OR CHUNK SIZE [**] [Classification: Unknown Traffic] [Priority: 3] {TCP} 10.14.141.30:51876
08/16-14:16:55.414617  [**] [139:1:1] (spp_sdf) SDF Combination Alert [**] [Classification: Senstive Data] [Priority: 2] {PROTO:254} 69.165.169.197 -> 10.14.141.30
08/16-14:18:01.640740  [**] [129:14:1] TCP Timestamp is missing [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 31.23.243.173:45778 -> 10.14.141.30:4
08/16-14:26:50.134766  [**] [129:13:1] 4-way handshake detected [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 24.43.69.30:49001 -> 10.14.141.30:444
08/16-14:32:56.378122  [**] [129:14:1] TCP Timestamp is missing [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 94.136.204.34:37341 -> 10.14.141.30:3
08/16-14:35:20.709858  [**] [139:1:1] (spp_sdf) SDF Combination Alert [**] [Classification: Senstive Data] [Priority: 2] {PROTO:254} 91.228.177.93 -> 10.14.141.30
08/10-23:45:52.644421  [**] [1:30918:1] BLACKLIST User-Agent known malicious user agent - User-Agent User-Agent Mozilla [**] [Classification: A Network Trojan was detec
08/16-06:51:57.322596 [**] [3:11672:8] BROWSER-OTHER Mozilla Network Security Services SSLv2 stack overflow attempt [**] [Classification: Attempted Administrator Privil
```

Fig 32. Generation of alerts using snort

6.10 Generation of Priority graph Using Python in Anaconda IDE.

```
import matplotlib.pyplot as plt; plt.rcdefaults()
import nump as np
import matplotlib.pyplot as plt
file=open("snot.txt","r")
priority=[ ]
count=[ ]
for line in file:
words=[words for words in line.split(",")]
 priority.append(int(words[2].strip("\n")))
for i in range(1,5):
 count.append(priority.count(i))
print(count)
values=np.arange(5)
count.sort()
print(count)
print(value)
plt.bar(values, count, align='center', alpha=0.5)
plt.ylabe('Thrad priority',i)
plt.title('Thread Priority Count')
plt.show()
```

7.Result and discussion

Network Forensics Evidence Generated with Snort and Wireshark

A network forensics investigation that entails the use of Snort involves three forms of data which the network forensics examiner must address within the court of law. The first form of data is the capturing or captured binary network sniffer data. During this stage, the network forensics examiner or the organization must prove that the gathered data was obtained using business record procedures (which include nontrained equipment). The second form of data, which occurs during the Preprocessor and Detection Engine Components stages, is the preprocessor and detection rule criteria used to identify the security intrusion or security violation. The final form of data is the IDS alerts generated and saved as a log file or in a database. The various forms of Snort generated evidence collected during network forensics investigations require the network forensics examiner organizations how to produce and handle digital or electronically generated evidence before the organization experiences a security incident, if possible. The teaching process entails making sure the organization understands the requirements for having the court accept evidence obtained during an investigation. As a result, the network forensics investigator must plan for and address this issue early on, before the collection of any must network based evidence within the organization. The network forensics examiner must ensure organizations are familiar with the four principles of network forensics evidence. The following is a list of the four principles:

1. Understanding the Life Cycle of Evidence
2. Adhering to the Rules of Evidence Criteria
3. Knowing the Uniqueness of Digital Evidence
4. Submitting of Computer Records

The first principle, Understanding the Life Cycle of Evidence, requires all parties involved in the investigation understand that evidence has different life cycle phases and everyone must properly follow each phase in accordance with sound forensics procedures. Figure shown below presents the five phases of the life cycle of evidence.

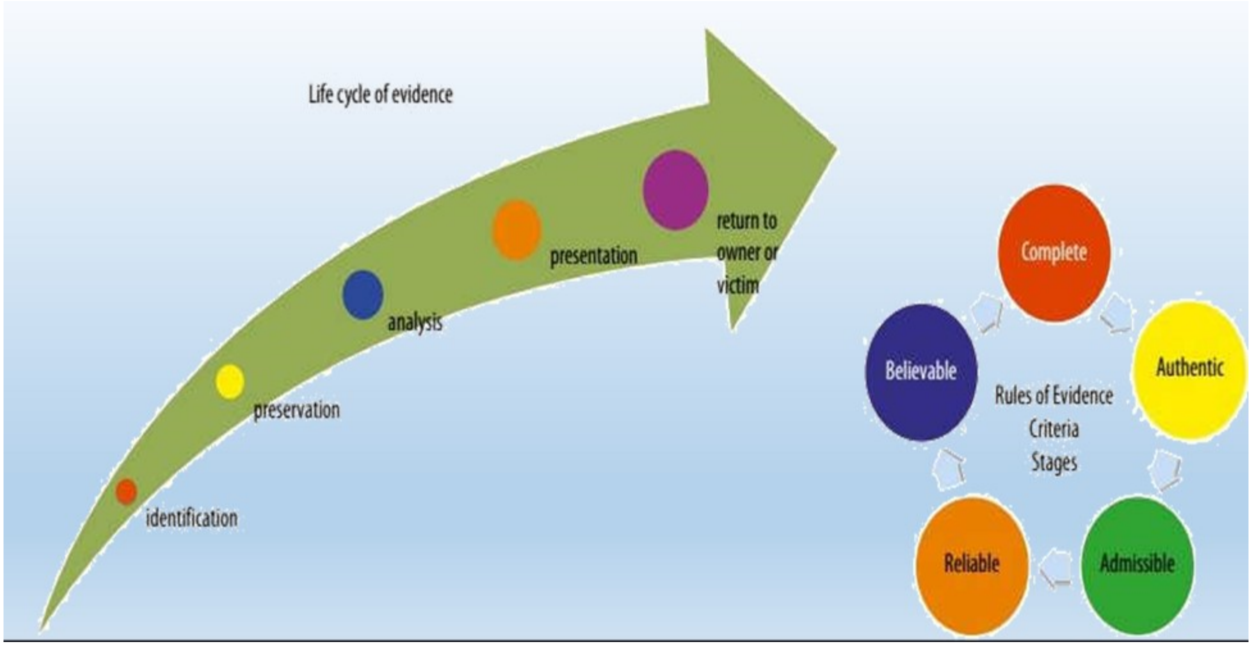


Fig 33. The Stages and Life cycle of evidence

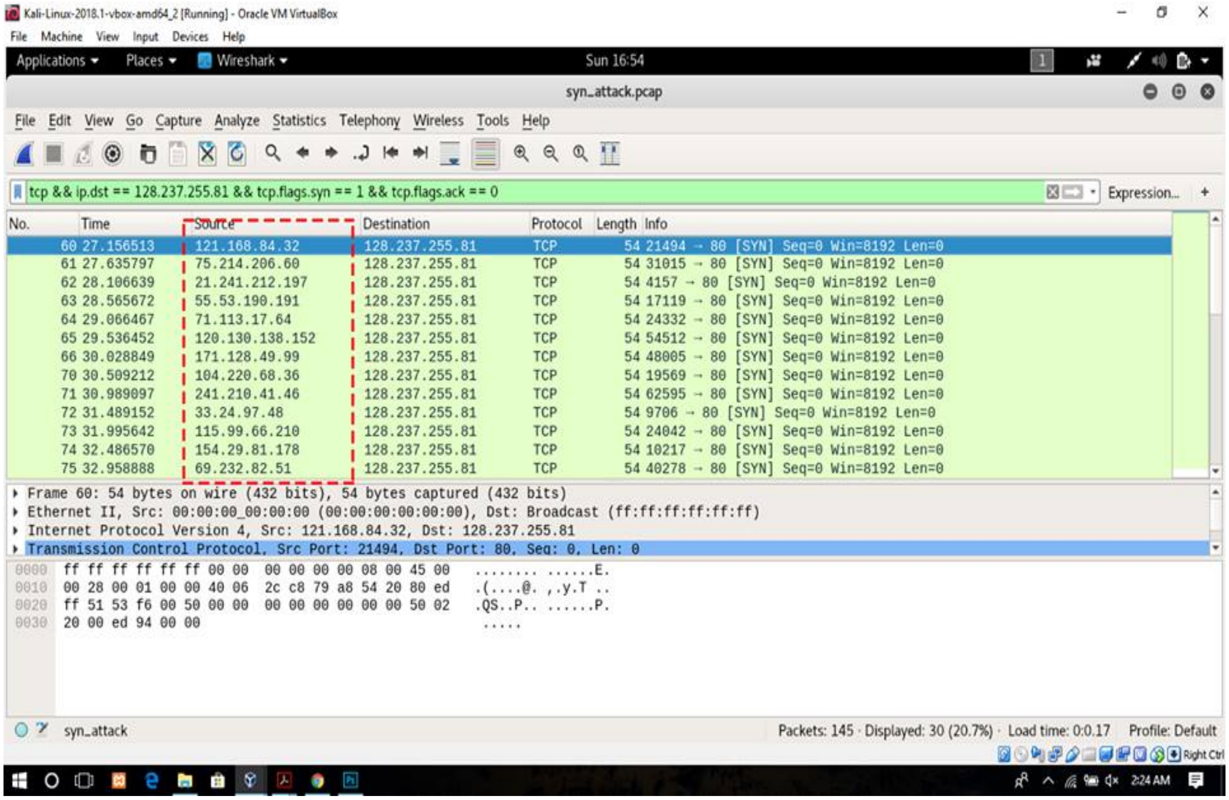
It appears a SYN-flood style DDoS has been carried out on this system. Send us a list of the IP addresses of the attackers (in any order, separated by spaces), so we can track them down and stop them.

First, we have to understand what SYN attack is. Simple is attacker send many packets with flag SYN = 1 at a time, server can't respond ACK because timer is longer than sending timer, server is overload.

Open Pcap file with the help of Wireshark, then go to statics and then conversations: Wireshark > Statics > Conversations.

Use filter to filt all packets from attack:

tcp && ip.dst == 128.237.255.81 && tcp.flags.syn == 1 && tcp.flags.ack == 0


Kali-Linux-2018.1-vbox-amd64_2 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Wireshark
Sun 16:54
syn_attack.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression...

```
tcp && ip.dst == 128.237.255.81 && tcp.flags.syn == 1 && tcp.flags.ack == 0
```

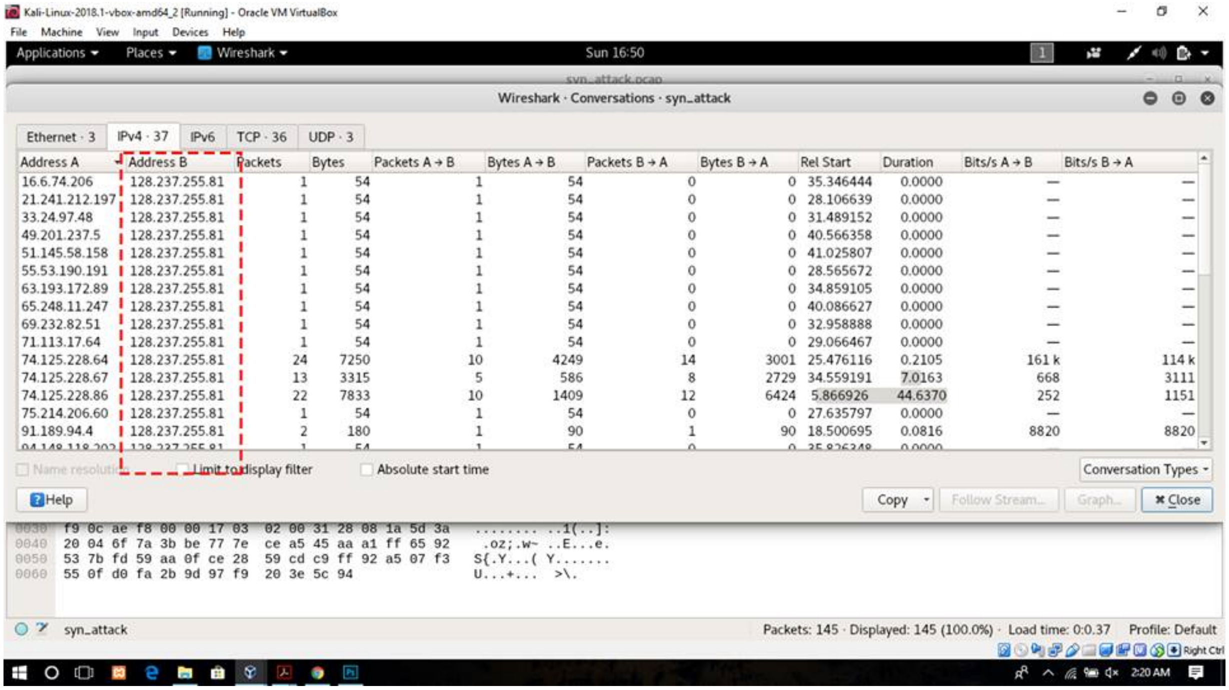
```
No.    Time         Source            Destination      Protocol  Length Info
    60 27.156513    121.168.84.32     128.237.255.81   TCP           54 21494 → 80 [SYN] Seq=0 Win=8192 Len=0
    61 27.635797    75.214.206.60     128.237.255.81   TCP           54 31015 → 80 [SYN] Seq=0 Win=8192 Len=0
    62 28.106639    21.241.212.197    128.237.255.81   TCP           54 4157 → 80 [SYN] Seq=0 Win=8192 Len=0
    63 28.565672    55.53.190.191     128.237.255.81   TCP           54 17119 → 80 [SYN] Seq=0 Win=8192 Len=0
    64 29.066467    71.113.17.64      128.237.255.81   TCP           54 24332 → 80 [SYN] Seq=0 Win=8192 Len=0
    65 29.536452    120.130.138.152   128.237.255.81   TCP           54 54512 → 80 [SYN] Seq=0 Win=8192 Len=0
    66 30.028849    171.128.49.99     128.237.255.81   TCP           54 48005 → 80 [SYN] Seq=0 Win=8192 Len=0
    70 30.509212    104.220.68.36     128.237.255.81   TCP           54 19569 → 80 [SYN] Seq=0 Win=8192 Len=0
    71 30.989097    241.210.41.46     128.237.255.81   TCP           54 62595 → 80 [SYN] Seq=0 Win=8192 Len=0
    72 31.489152    33.24.97.48       128.237.255.81   TCP           54 9706 → 80 [SYN] Seq=0 Win=8192 Len=0
    73 31.995642    115.99.66.210     128.237.255.81   TCP           54 24042 → 80 [SYN] Seq=0 Win=8192 Len=0
    74 32.486570    154.29.81.178     128.237.255.81   TCP           54 10217 → 80 [SYN] Seq=0 Win=8192 Len=0
    75 32.958888    69.232.82.51      128.237.255.81   TCP           54 40278 → 80 [SYN] Seq=0 Win=8192 Len=0
```

```
▸ Frame 60: 54 bytes on wire (432 bits), 54 bytes captured (432 bits)
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
▸ Internet Protocol Version 4, Src: 121.168.84.32, Dst: 128.237.255.81
▸ Transmission Control Protocol, Src Port: 21494, Dst Port: 80, Seq: 0, Len: 0
```

```
0000  ff ff ff ff ff ff 00 00  00 00 00 00 08 00 45 00   ........ ......E.
0010  00 28 00 01 00 00 40 06  2c c8 79 a8 54 20 80 ed   .(....@. ,.y.T ..
0020  ff 51 53 f6 00 50 00 00  00 00 00 00 00 00 50 02   .QS..P.. ......P.
0030  20 00 ed 94 00 00                                   .....
```

syn_attack   Packets: 145 · Displayed: 30 (20.7%) · Load time: 0:0.17   Profile: Default

Fig 34. Flooding Source IP using Wireshark


Kali-Linux-2018.1-vbox-amd64_2 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Wireshark
Sun 16:50
syn_attack.pcap
Wireshark · Conversations · syn_attack
Ethernet · 3
IPv4 · 37
IPv6
TCP · 36
UDP · 3

Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
16.6.74.206	128.237.255.81	1	54	1	54	0	0	35.346444	0.0000	—	—
21.241.212.197	128.237.255.81	1	54	1	54	0	0	28.106639	0.0000	—	—
33.24.97.48	128.237.255.81	1	54	1	54	0	0	31.489152	0.0000	—	—
49.201.237.5	128.237.255.81	1	54	1	54	0	0	40.566358	0.0000	—	—
51.145.58.158	128.237.255.81	1	54	1	54	0	0	41.025807	0.0000	—	—
55.53.190.191	128.237.255.81	1	54	1	54	0	0	28.565672	0.0000	—	—
63.193.172.89	128.237.255.81	1	54	1	54	0	0	34.859105	0.0000	—	—
65.248.11.247	128.237.255.81	1	54	1	54	0	0	40.086627	0.0000	—	—
69.232.82.51	128.237.255.81	1	54	1	54	0	0	32.958888	0.0000	—	—
71.113.17.64	128.237.255.81	1	54	1	54	0	0	29.066467	0.0000	—	—
74.125.228.64	128.237.255.81	24	7250	10	4249	14	3001	25.476116	0.2105	161 k	114 k
74.125.228.67	128.237.255.81	13	3315	5	586	8	2729	34.559191	7.0163	668	3111
74.125.228.86	128.237.255.81	22	7833	10	1409	12	6424	5.866926	44.6370	252	1151
75.214.206.60	128.237.255.81	1	54	1	54	0	0	27.635797	0.0000	—	—
91.189.94.4	128.237.255.81	2	180	1	90	1	90	18.500695	0.0816	8820	8820
94.148.118.202	128.237.255.81	1	54	1	54	0	0	35.826348	0.0000	—	—

□ Name resolution   □ Limit to display filter   □ Absolute start time

```
0030  f9 0c ae f8 00 00 17 03  02 00 31 28 08 1a 5d 3a   ........ ..1(..]:
0040  20 04 6f 7a 3b be 77 7e  ce a5 45 aa a1 ff 65 92    .oz;.w~ ..E...e.
0050  53 7b fd 59 aa 0f ce 28  59 cd c9 ff 92 a5 07 f3   S{.Y...( Y.......
0060  55 0f d0 fa 2b 9d 97 f9  20 3e 5c 94               U...+...  >\.
```

syn_attack   Packets: 145 · Displayed: 145 (100.0%) · Load time: 0:0.37   Profile: Default

Fig 35. Server (victim): 128.237.255.81

By using python in Anaconda IDE, we will generate the attack graph and the Thread Priority Count graph to detect the malicious packet from the networked environment, In a network environment some of the addresses try to send maximum number of SYN alerts and gives us the maximum priority for flooding on a network environment and which is used to declare as the malicious packet or untrusted address.

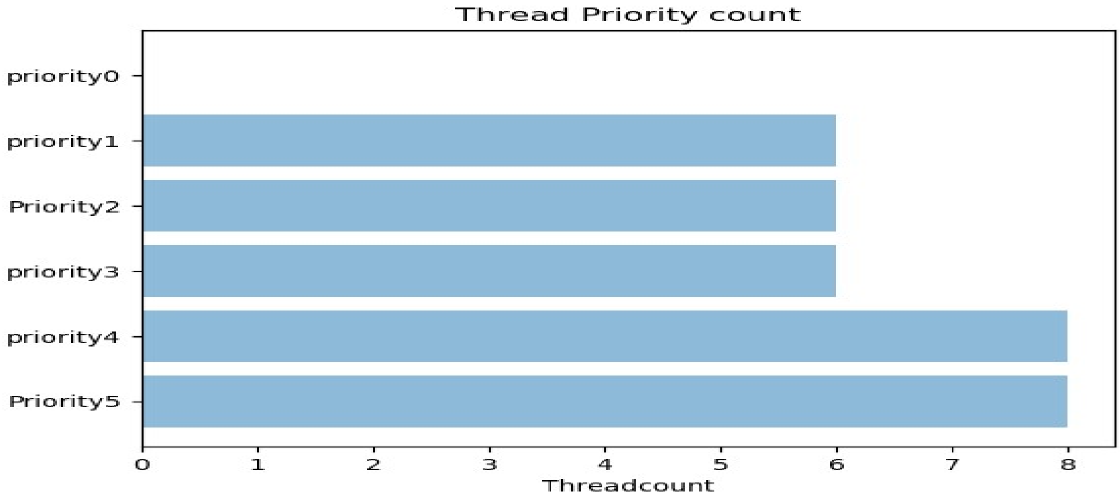


Fig 36. Thread Priority Graph

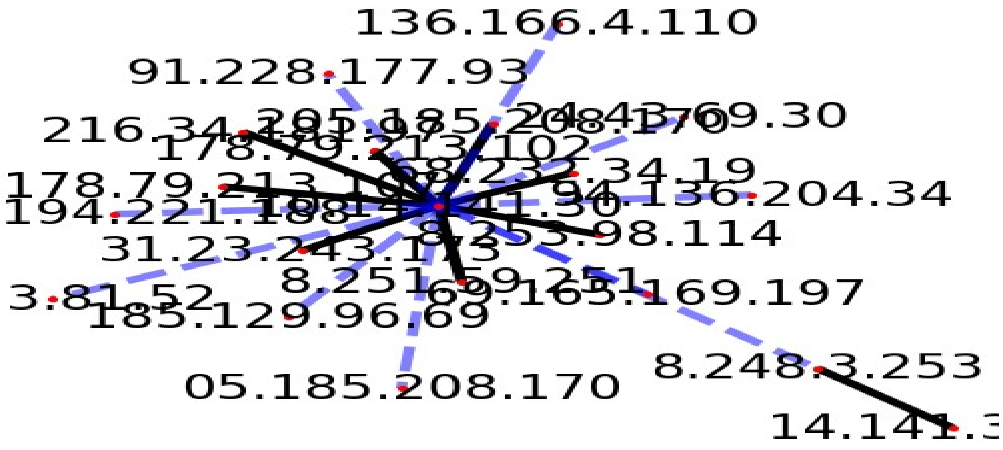
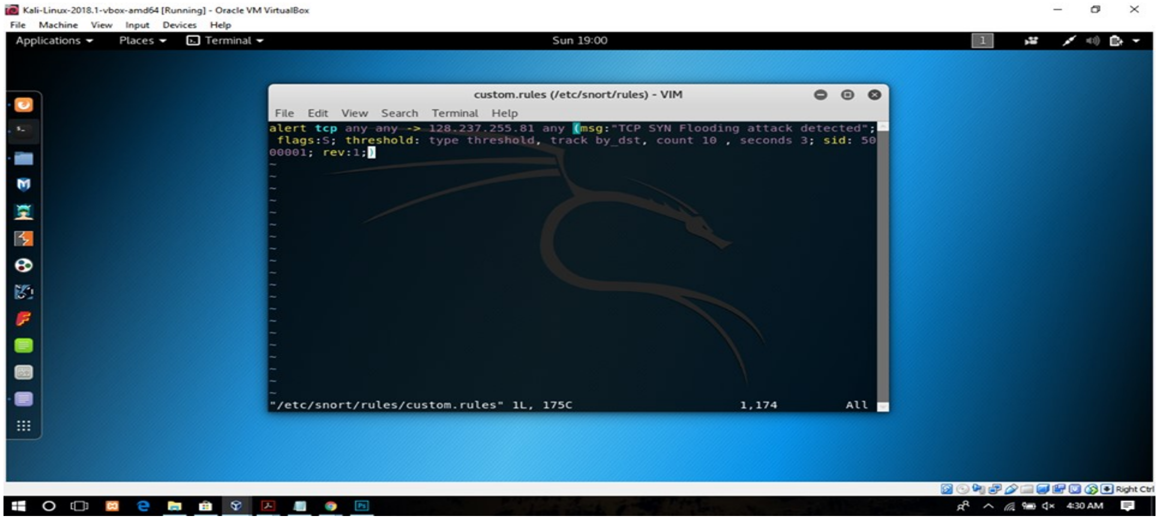


Fig 37. Attack Graph for network forensic analysis

Custom rule to detect SYN-FLOOD ATTACK

By generating the snort rule to detect the TCP SYN FLOOD ATTACK.

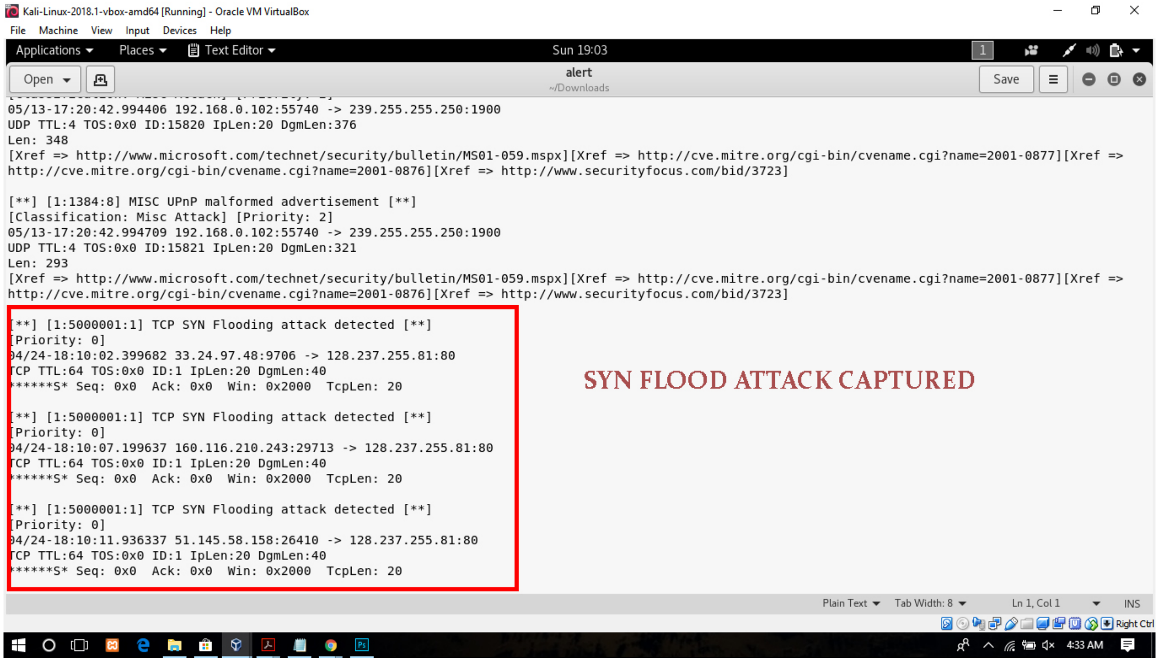
alert tcp any any -> 128.237.255.81 any (msg:"TCP SYN Flooding attack detected"; flags:S; threshold: type threshold, track by_dst, count 10 , seconds 30; sid: 5000001; rev:1;)



Kali-Linux-2018.1-vbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Terminal
Sun 19:00
custom.rules (/etc/snort/rules) - VIM
File Edit View Search Terminal Help

```
alert tcp any any -> 128.237.255.81 any (msg:"TCP SYN Flooding attack detected";
 flags:S; threshold: type threshold, track by_dst, count 10 , seconds 3; sid: 50
00001; rev:1;)
"/etc/snort/rules/custom.rules" 1L, 175C                        1,174         All
```

Fig 38. Snort Rule for TCP SYN FLOOD



Kali-Linux-2018.1-vbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Text Editor
Sun 19:03
Open
alert
~/Downloads
Save
Plain Text Tab Width: 8 Ln 1, Col 1 INS

```
05/13-17:20:42.994406 192.168.0.102:55740 -> 239.255.255.250:1900
UDP TTL:4 TOS:0x0 ID:15820 IpLen:20 DgmLen:376
Len: 348
[Xref => http://www.microsoft.com/technet/security/bulletin/MS01-059.mspx][Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2001-0877][Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=2001-0876][Xref => http://www.securityfocus.com/bid/3723]

[**] [1:1384:8] MISC UPnP malformed advertisement [**]
[Classification: Misc Attack] [Priority: 2]
05/13-17:20:42.994709 192.168.0.102:55740 -> 239.255.255.250:1900
UDP TTL:4 TOS:0x0 ID:15821 IpLen:20 DgmLen:321
Len: 293
[Xref => http://www.microsoft.com/technet/security/bulletin/MS01-059.mspx][Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2001-0877][Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=2001-0876][Xref => http://www.securityfocus.com/bid/3723]

[**] [1:5000001:1] TCP SYN Flooding attack detected [**]
[Priority: 0]
04/24-18:10:02.399682 33.24.97.48:9706 -> 128.237.255.81:80
TCP TTL:64 TOS:0x0 ID:1 IpLen:20 DgmLen:40
******S* Seq: 0x0  Ack: 0x0  Win: 0x2000  TcpLen: 20

[**] [1:5000001:1] TCP SYN Flooding attack detected [**]
[Priority: 0]
04/24-18:10:07.199637 160.116.210.243:29713 -> 128.237.255.81:80
TCP TTL:64 TOS:0x0 ID:1 IpLen:20 DgmLen:40
******S* Seq: 0x0  Ack: 0x0  Win: 0x2000  TcpLen: 20

[**] [1:5000001:1] TCP SYN Flooding attack detected [**]
[Priority: 0]
04/24-18:10:11.936337 51.145.58.158:26410 -> 128.237.255.81:80
TCP TTL:64 TOS:0x0 ID:1 IpLen:20 DgmLen:40
******S* Seq: 0x0  Ack: 0x0  Win: 0x2000  TcpLen: 20
```

SYN FLOOD ATTACK CAPTURED

Fig 39. Alert generated for SYN FLOOD ATTACK

9. Conclusion and future scope

In this work we have tried to improve the detection rate of network based attacks using the new community rules. After using SNORT tool as an intrusion detection system, we have seen it has the full ability to detect all attacks which have predefined rule for signature match. Today's hackers are very clever and they generate the new signature for different attacks, so they may be success full some time but not always, because we have a weapon like rules and plugging in snort. Our purpose for implementing this project is, detecting stealthy TCP, ICMP, UDP packets which are malicious in snort. In result part we have seen that snort is able to detect attacks and it can easily generate alerts on the basis of rules configured during the installation process. Therefore, in order to detect and generate alerts on the basis of rule based intrusion detection method we have to use the latest community rules such that we can generate for zero days based network attacks also.

In future, we plan to create an automated solution where we can detect those attacks for which there is no rule mentioned in the configuration file. With the help of machine learning concept, Using the alert generation file as a training data set we will train our platform to detect the zero days based network attacks. We also plan to integrate various methods of network intrusion detection (i.e. rule based, Anomaly based, Behaviour based) on the same data and observe whether the integrated approach performs better than the individual approach or not.

10. References/Bibliography

1. SULAIMON ADENIJI ADEBAYO, “Network Security-Attacks and Mitigation”, Turku University of Applied Science, 2012. Accessed: May 2017.
2. Network Forensics 101: Finding the Needle in the Haystack, Wild packet Publisher. Accessed: May 2017.
3. Marcus Ranum, “Network Forensic Analysis Definition”. Source: www.wikipedia.com/network forensics. Accessed: May 2017.
4. Hakan Semerci,” **Analysis of Intrusion Prevention Methods**” Izmir Institute Of Technology,2004. Accessed: May 2017.
5. ARACELI BARRADAS-ACOSTA, ELEAZAR AGUIRRE ANAYA, MARIKO NAKANO-MIYATAKE, HECTOR PEREZ-MEANA,” **Neural Network Based Attack Detection Algorithm**”,WSEAS transactions on Computers,2013.
6. Ioannis A. Apostolakis, “Network Forensics: Problems and Solutions”, E-Democracy: Challenges of the Digital Era, At Athens, Greece, Volume: p.307-318,2006.
7. Yusuf Bhaiji,”Understanding and Preventing Against Layer 2 attack”. Source: blog.cisco.com. Accessed: May 2017.
8. Mirkovic, Jelena, and Peter Reiher. "A taxonomy of DDoS attack and DDoS defense mechanisms." *ACM SIGCOMM Computer Communication Review* 34.2 (2004): 39-53.2004.
9. Pelaez, Jaun C., and Eduardo B. Fernandez. "Network Forensic Models for Converged Architectures." *International Journal on Advances in security* 3.1 (2010).
10. Prosise, Chris, Kevin Mandia, and Matt Pepe. "Incident response & computer forensics." (2003).
11. S. Krasser, G.conti and J.Gizzard “Real-time and forensic network data analysis using animated and coordinated visualization[2005].
12. AK Kaushik, R.C. Joshi, Emmanuel S.pilli “Network forensic system for port scanning attack.
13. AK Kaushik, R.C. Joshi, Emmanuel S.pilli “Network forensic system for ICMP attack.
14. Ali Reza Arasteh, Mourad Debbabi “Analyzing multiple logs for forensic evidence”.
15. Snort- light weight intrusion detection for networks by Martin Roech.

10. List of Figures