International Journal of Computer Networks and Communications Security

VOL. 3, NO. 2, FEBRUARY 2015, 33–42
Available online at: www.ijcncs.org
E-ISSN 2308-9830 (Online) / ISSN 2410-0595 (Print)

Network Forensics: Detection and Analysis of Stealth Port Scanning Attack

Rajni Ranjan Singh[1] and Deepak Singh Tomar[2]

[1, 2] Maulana Azad National Institute of Technology, Bhopal, India

E-mail: [1]ranjansingh06@gmail.com, [2]deepaktomar@manit.ac.in

ABSTRACT

Network administrator performs port scanning for the purpose of network monitoring and troubleshooting on the other hand this facility become vulnerability when attacker performs port scanning for probing networks, searching for vulnerabilities and then infiltrate IT assets. It is often a primarily tactic that is adopted by attacker prior to launching a targeted cyber-attack. Moreover in recent times, port scanning techniques become highly distributed, composite, hybrid, and stealthy, therefore almost all current detection techniques are unfeasible. Stealth is considered to be a type of port scan which is undetected by available auditing tools such as firewall, routes, filters etc. A stealth port scan method does not produce any TCP sessions; hence, none of these scans should appear in any of the application logs. Therefore, it is of vital importance to research and adopt methods for the detection and attribution of stealth port scanning attack. In this work a network forensic architecture for detection and analysis of stealth port Scanning attack is proposed. It consist of two main modules, a capturing module that captures fine grained evidences from the network traffic and an analysis module that classifies each packet based on the predefined signature. A proof of concept prototype is implemented which utilize, operational network traffic data injected with crafted scans to test the system. It is reported that the proposed system correctly identifies crafted scans injected into real traffic.

Keywords: *Network forensic, Network Scanning, Port scan, NIDS, Snort.*

1 INTRODUCTION

Port scanning is a growing network security concern due to the fact that it is the primary stage of an intrusion attempt that enables the attacker to remotely locate, target and subsequent exploit vulnerable systems. It is observed that 50% of attacks against cyber systems are preceded by some form of network scanning activity [1].

Hence the capability to detect and attribute various scanning activity is a very important task to achieve as this will prevent the actual cyber-attack from occurring.

Network forensics is the field of applying forensic science to the computer networks in order to discover the source of network crimes. The objective of network forensic is to identify malicious activities from the traffic logs, discover their details, and to assess the damage. [2]

Garfinkel et .al [3] categorize the network forensic systems into two types, based on their collection characteristics:

'Catch-it-as-you-can' Systems where all the packets passing through a specific traffic point are captured and analysis is subsequently. This system requires large amounts of storage.

'Stop-look- and listen' Systems where each packet is analyzed in memory and certain information is saved for future analysis. This system requires faster processor.

Network forensic investigation of port scanning attack is more challenging because of following reasons.

a) Port scan attack traffic is seems to be very much similar to the conventional network traffic. Thus bifurcation between attack and traditional traffic can be a difficult task.

b) Port scan can be slow (one packet/hour or packet/day) therefore storage capability is another factor in collecting large amounts of live network evidence. High-speed networks (Gigabit Ethernet, SONET speeds and higher, etc.) cannot realistically be captured in their entirety beyond a certain length of time.

c) Port scanning is a precursor of attack so it is needed to discover it as early as possible in order to block actual attack from occurring thus fast analysis system is required.

d) Some form of port scanning attack (Stealth) is unobserved by conventional networking devices like router, firewall etc thus it's needed to put specialized devices for traffic collection task.

2 PORT SCANNING

Port scanning is a technique utilizes by Intruders and network administrators to gather information from computers connected to a network. Network administrators utilize port scans to recognize open ports of a system so that they may be able to limit access to those ports, or shut them off completely. Intruders use port scanning in the same way that administrators do, but with malicious intention.

Port scanning is classified in three categories based on the different types of packets used for scanning [4, 8].

a. Full open: This type of scan method utilizes connect () call functionality for initiating a full connection to a remote host using a typical three-way TCP/IP handshake. Its limitation is that all the scanning attempts are logged by the destination application. It is easily detected when connection logs are examined.

b. Half-open scanning: The term 'half-open' applies to the way the client terminates the connection before the three-way handshake is completed. As such, these scan methods often go unlogged by the destination application. Since the half open scan techniques uses known TCP flags, it can be detected by an edge firewall rather easily.

c. Stealth scanning: Stealth is considered to be any scan that bypassing filters, firewalls, routers and appearing as casual network traffic. Most common stealth scan is conducted by setting individual flags (ACK, FIN, RST) or NULL/ All flags of packets. Stealth scanning techniques are discussed here.

(1) SYN/ACK Scan: It is relatively fast scan method that avoids the use of three way handshake. In this scan type source sends a SYN with ACK flag to the target. For a closed port, the target will replay with a RST packet (A TCP packet with reset flag set) while a request to an open port will not generate a response. This scan technique generate notable amount of false positives due to the filtering devices, heavy traffic, slow link, and timeouts etc.

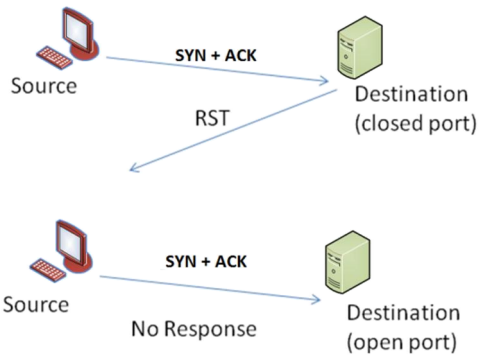


Fig. 1. SYN/ACK Scanning Packet Exchange

(2) XMUS, NULL AND FIN Scan: These three port scanning techniques are grouped together because their individual functionality is very similar. They send a single frame to a TCP port without any TCP handshaking or any additional packet transfers.

As per the RFC 793[9] if a packet is sent to the open ports without the SYN, ACK and RST bits set (Any combination of the other three FIN, PSH, and URG are OK) than receiver drops the packet and no response returned. If the destination port is closed then RST is returned. Details of individual scan are given in this section.

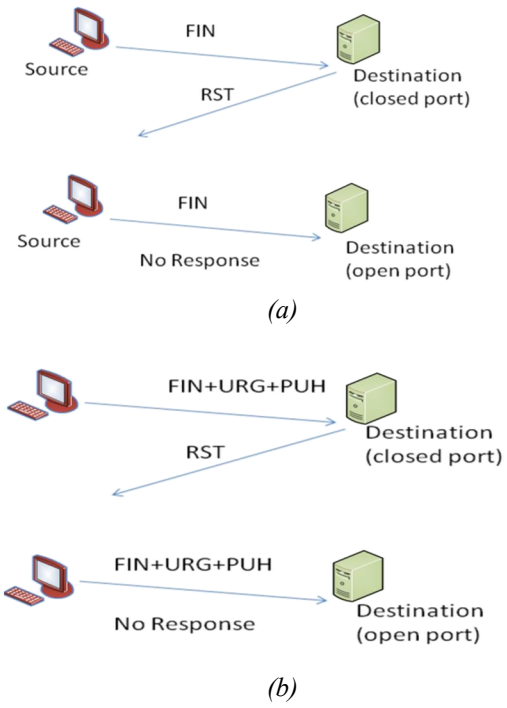
XMAS Scan - XMAS scans send a packet with the FIN, URG, and PSH flags set. If the port is open, there is no response returned but if the port is closed, the target responds with a RST packet. Note that, the Xmas Tree scan takes its name from the flags allied to (00101001), which become visible similar to the lights of a Christmas tree.

FIN Scan - A FIN scan is similar to an XMAS scan but sends a packet with just the FIN flag set (remaining URG and PUH bit are not set). FIN scans receive the same response and have the same characteristics as XMAS scans.

NULL Scan - A NULL scan is also alike to XMAS and FIN in its limitations and response, but it just sends a packet with no flags set.

The main strength of these scan types is that they can pass through certain non-stateful firewalls and packet filtering routers and does work against most Unix-based systems. For identifying a closed port, only two packets are transferred. On the other hand, just a single packet is required to identify an open port. It does not produce any TCP sessions; hence, none of these scans should appear in any of the application logs.

However, these scan are not able to differential between open and closed port for Microsoft machines. On the other hand by using these scans, if an open port found indicates that the target machine is not from Microsoft OS family.



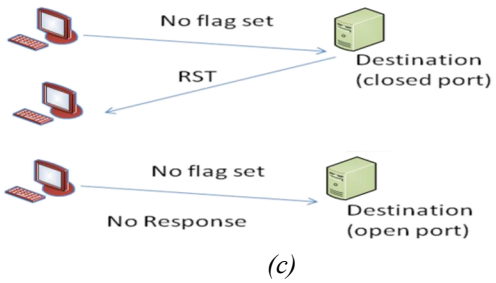


Fig. 2. FIN (a), XMUS (b) and NULL (c) Scan Packet Exchange.

(3) ACK Scan: ACK scan is intended to identify ports that are filtered through firewall. It can never identify open ports. An ACK scan operates by sending a TCP ACK frame to a remote port. If there are no responses or an ICMP destination unreachable message is returned, then the port is considered to be filtered. If the remote port returns a RST packet, the remote target is categorized as unfiltered.

Some tcp stacks returned windows size when responding to an ACK packet. if destination port is closed then returned window size become non zero and in the case of open port the window size become zero. However window size not returned by all devices.

(4) TCP Fragmenting Scan: it is designed to evade packet filters. This scan types may apply any of the over mentioned techniques the only difference is that while sending a probe packet it first decomposes into many fragments. This technique is effective since packet filters or intrusion detection systems do not buffer the entire set of packets due to performance issues.

3 PROPOSED NETWORK FORENSIC SYSTEM ARCHITECTURE

The proposed system architecture is the generalized architecture, which is extended to detect and analyze stealth port scanning attack.

The system is developed to capture network traffic from network interface of acquisition system. Each captured packet first examined by the marking module, which marks it as relevant or irrelevant based on the flag values in the packet. Only the packet marks as relevant goes to the analysis module.

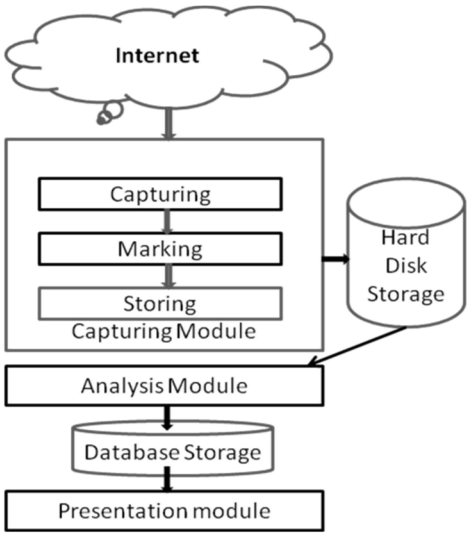


Fig. 3. Network Forensic System Architecture

Analysis module is responsible for identifying scanning attack patterns. If pattern found then source IP address is classified as suspicious and as normal otherwise and finally produce valuable information of all identified scanning attempts.

The work carried out in this paper mainly focused on the detection and analysis of XMUS, NULL and FIN scanning attack. These scans are the most commonly used stealth port scanning methods.

The Architecture is broadly developed into three modules: capturing module, analysis module, and presentation module. Modules description and implementation details are described in the following sections.

3.1 Capturing module

This module insure that only the actual packets utilized for port scanning attack are stored .This results is less amount of storage required as compared to the normal process which captures all the packets passing through network interface and stores them on to the system.

Acquisition system should be deployed at the minimum distance position between source and destination in order to reduce error rate. Increasing distance may affect overall network latency, causing timeouts, signal strengths and errors. Distance affects the completeness of data collection. Therefore it is effective to deploy acquisition system at the edge of the network, rather than deploying after switches or with in particular subnet.

It is better to deploy two independent instances of capturing system for collecting evidences from the same live network source during the same acquisition time window. It provides more complete data collection, reduces error and increase readability. After collection both parties show identical set of captured network traffic this corroboration shows independent verification of evidence collection.

This module is further divided into three sub modules: capturing, Marking and storing.

A. Capturing:

Packet capturing is the act of collecting data as it travels over a network. Every time a network card collects an Ethernet frame it verifies that its destination MAC address matches its own. If it does, it generates an inter¬rupt request. The card driver routine is responsible for handling the interrupt. The driver timestamps received data and cop¬ies it from the card buffer to kernel space memory. Then, it decides which type of packet has been received looking at the ether type field of the Ethernet header and passes it to the appropriate protocol handler in the protocol stack.

When sniffer is utilized for packet capturing, the card driver also sends a copy of any received or transmitted packet to the packet filter.

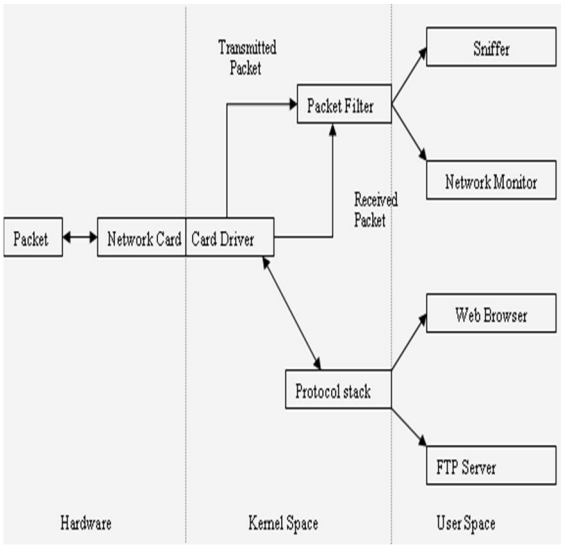


Fig. 4. Packet Capturing Process

Open source sniffer software's is explored here for the capturing of network packets.

1) Marking:

This module is to marks the incoming packets as relevant or irrelevant for further analysis. As over mentioned that if a incoming TCP packet without the SYN, ACK and RST bits set are the prima facie

evidence for FIN, XMUS, NULL scan, therefore mark these packets as relevant. The steps are listed as follows.

1) If the protocol used in the packet is not TCP then mark the packet as irrelevant and break, else go to step 2.
2) If any of the flag SYN, RST or ACK is set then mark the packet as irrelevant and break, else go to step3.
3) Mark the packet as relevant.
4) Repeat the steps for each captured packet.

Pseudo code for capturing process is shown in the Figure 5. Relevent_Packet_Vector is a data repository that stores relevant packets

```
Variables used:
# Packet i (flag): return the flag value
# Packet i (Pro): return the protocol
---------------------------------------------------------------------------
 Input: Incoming packets
Output: Relevent_Packet _Vector.
---------------------------------------------------------------------------
1. Initialize:
    Relevent_Packet _Vector [p1,p2……pn] -> [0, 0….0];
2. Process incoming packet
3. if (Packet i (Pro)= =TCP) then go to step 4 else go to step2.
4. if (Packet i (flag) ==SYN OR ACK OR RST ) then go to
   step 2 else go to step 5
5. Relevent_Packet _Vector. ->Packet i/* Add packet to
  vector*/ go to step 2.
```

Fig. 5. Pseudo code for packet capturing process

C. Storing:

The task of this module is to store the relevant packet into the local hard disk of the acquisition system. A MD5 hash of all the captured data is computed. It gives assurance that digital evidence has not been altered since it was obtained. Secondary copy of the captured data is supplied to the analysis module and the original collected network traffic is preserved on the write once, read only device.

3.2 Analysis Module

The captured packets are classified based on the proposed signature of XMUS, NULL and FIN scanning attack and if found than packet is assigned to the respective data repositories. Data repository named, XMUS_Scanning_Vector, FIN_Scanning_Vector and NULL_Scanning_Vector are developed to store classified packets of each scanning variants. The steps are listed as follows.

1) If no flags are set in the packet then assign it to NULL_Scanning_Vector and break, else go to step 2
2) If all URG, FIN and PSH flags are set then assign it to the XMUS_Scanning_Vector, else go to step 3.
3) If FIN flag is set and URG and/or PSH flag are not set then store it to the FIN_Scanning_Vector.
4) Repeat the above steps for each packet of Relevent_Packet_Vector

Pseudo code for packet analysis is given in Figure 6.

```
Input. Relevent_Packet _Vector
Output: . XMUS_Scanning_Vector,
FIN_Scanning_Vector and NULL_Scanning_Vector
1. Initialize:
  XMUS_Scanning_Vector [p1,p2……pn] - > [0, 0….0];
  FIN_Scanning_Vector [p1,p2……pn]   - > [0, 0….0];
  NULL_Scanning_Vector [p1,p2……pn]  - > [0, 0….0];
2. Process incoming packets in sequence i= 1, 2, 3…N,
    until all packet processed.
3. if (Packet i (flag) = =NULL ) then go to step 4 else go
   to step 5
4. NULL_Scanning_Vector->Packet i/* Add packet to
   vector*/
5. if ( Packet i (flag) = =FIN && URG && PUSH ) then
   go to step 6 else go to  step 7
6. XMUS_Scanning_Vector -> packet i /* Add packet to
   vector*/
7. if ( (Packet i (flag) = =FIN ) && (Packet i (flag)
   !=URG OR PUSH) then go to step 8 else go to step 2
8. FIN_Scanning_Vector  -> Packet i  /* Add packet to
    vector*/ go to step 2.
```

Fig. 6. Pseudo code for traffic analysis

The overall classification process of network packets is shown in the figure 7.

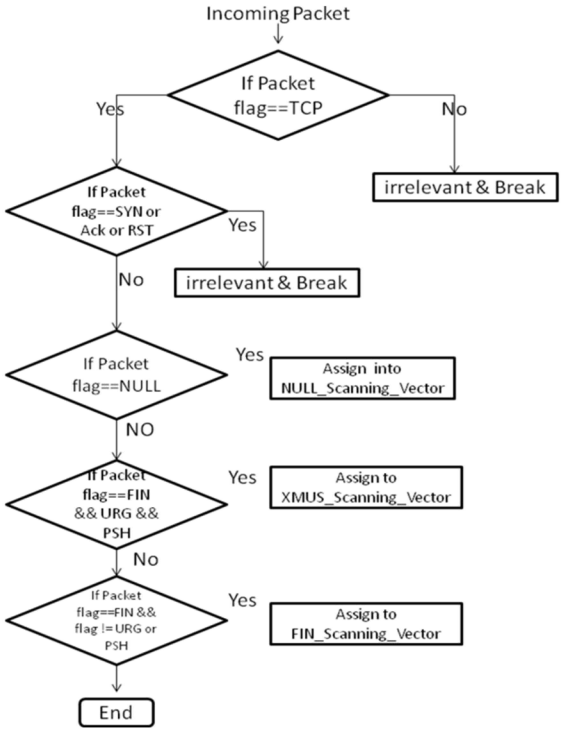


Fig. 7. A flow chart of packet analysis process

6 PRESENTATION MODULE

Packets stored in the repositories are updated to the database. Here fast query-driven validation is performed to identify whether the port scanning attack is carried out. Identified machines are marked as suspicious and as normal otherwise.

7 IMPLEMENTATION

SNORT is configured to implement the network forensic system. It is a free and open source network IDS, capable to perform real-time traffic examination and packet logging on Internet Protocol networks [10].

Snort can be configuring in three main modes: sniffer, packet logger, and network intrusion detection system (NIDS). NIDS mode is configurable, allowing Snort to look at network traffic for matches against a user defined rule set shown in the figure 6.

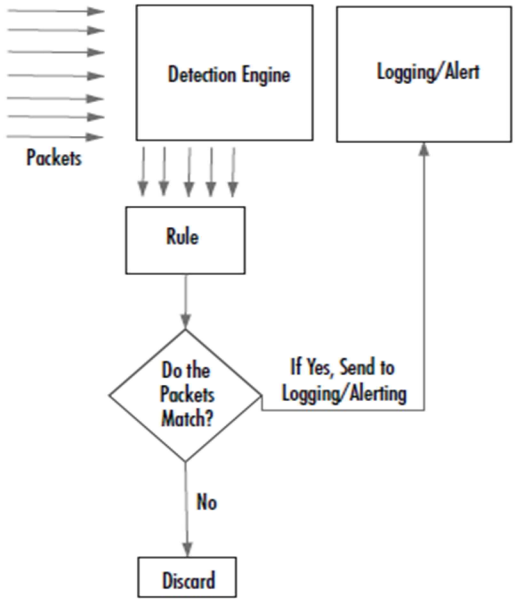


Fig. 8. Snort Detection Engine in NIDS mode

Packet capturing and marking performed by the snort IDS, which is configured in NIDS mode (option –c is utilize to activate NIDS mode). The marking process is carried out by the rule 1(shown in the figure 8). It is develop and configure to capture TCP packets, sent from any source and destined to any destination, without SYN, RST and ACK flags set.

Rule 1
alert tcp any any <> any any(flags:!SRA;sid:7987659;)
Rule 2
alert tcp any any <> any any (msg:"NULL Scanning Detected";flags:0;sid:7987660;)
Rule 3
alert tcp any any <> any any (msg:"XMUS Scanning Detected";flags:FPU;sid:7987660;)
Rule 4
alert tcp any any <> any any(flags:*FPU;sid:7987660;)
Rule 5
alert tcp any any <> any any (msg:"FIN Scanning Detected";flags:!PU;sid:7987660;)

Fig. 9. Snort Configured rule set

Each captured packet is compared against configured rule and if match found then packet is stored into the destination log folder of local hard drive and an alert entry is made into the alert file. A snort alert (in fast alert mode) has following structure.

<Timestamp><Alert message (configurable through rules)><Source and destination IP addresses> <Source and destination ports>
Example: 11/27-11:50:54.154077[**] [1:7987659:0] Scanning Detected [**] [Priority: 0] {TCP} 172.16.0.105:60033 -> 172.16.0.254:3306

Tcpdump file built during capturing process is supplied to the analysis module for identifying scanning packets. Here an offline analysis implementation is presented using Snort IDS. Snort <– r> option is applied to capture the traffic from tcpdump file.

Rule 2 and 3 (shown in the figure 7) is develop and configured to filter NULL and XMUS scan packets respectively. Rule shows that, the TCP packet from any source to any destination with no flag set or FIN, PUSH and URG flag is captured.

To identify Fin scanning packets, rule 4 and 5 has been applied. At first level rule 4 is applied to capture the packets having any of the FIN, PUSH, and URG flag set. Captured packets again processed by the snort using rule 5. It captures packets having FIN flag set.

Alert files created during analysis have been integrated, normalized and updated to the database for fast query based result analysis. MYSql [11] data repository has been utilized for high performance query driven result visualization.

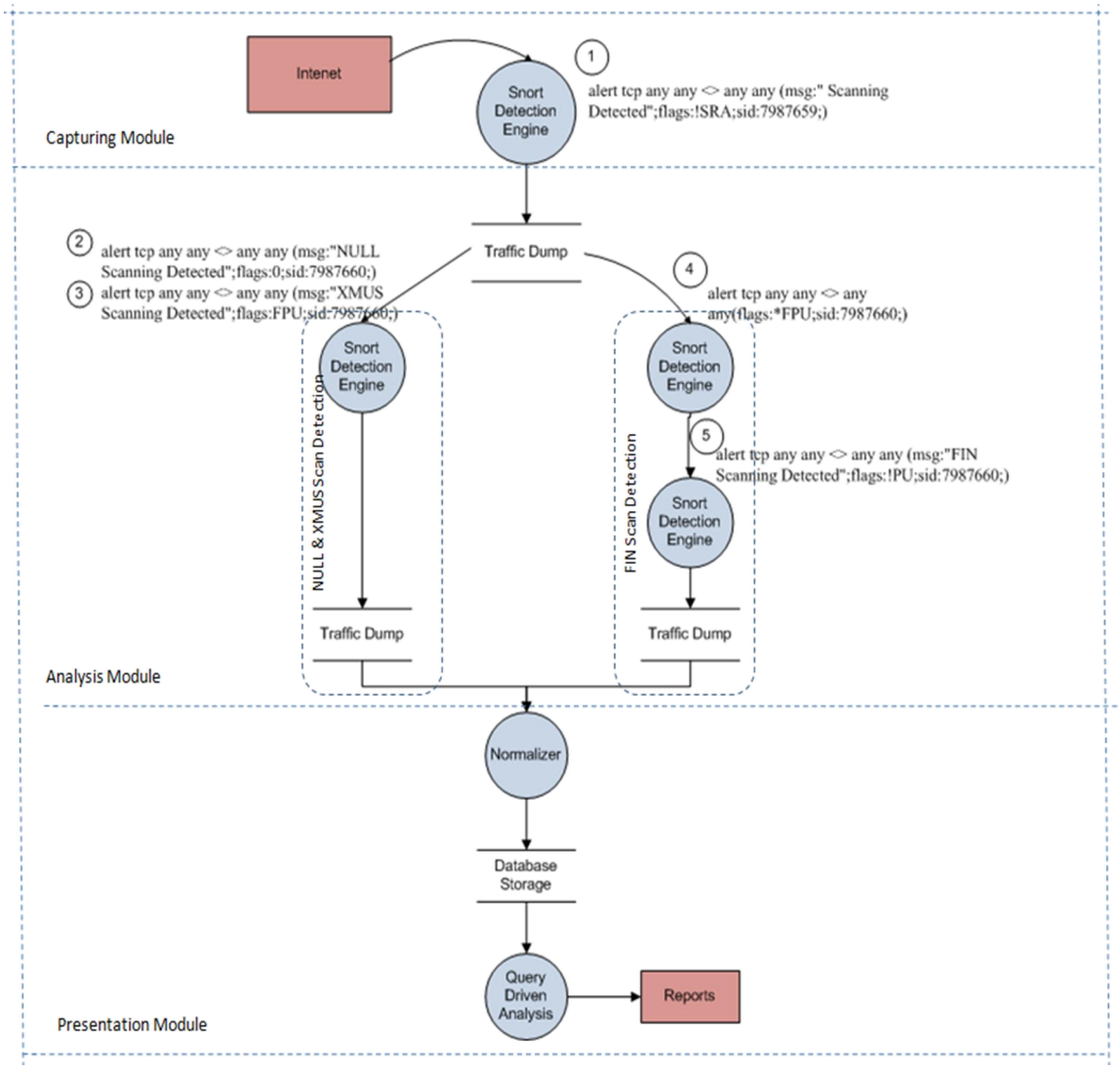


Fig. 9. Data Flow Diagram of Network Forensic system

8 EXPERIMENT AND RESULT

Proposed architecture has been implemented and tested on a small local area network. A test bed is built comprised of a victim machine. A switch (layer 2) and three other machines are scanners as shown in figure 10.

Scanners 1, 2 and 3 are equipped with nmap [12] performing FIN, XMUS and NULL scanning targeted to the victim machine. Table 1 shows the configuration summary of all the machines involves in the test bed.

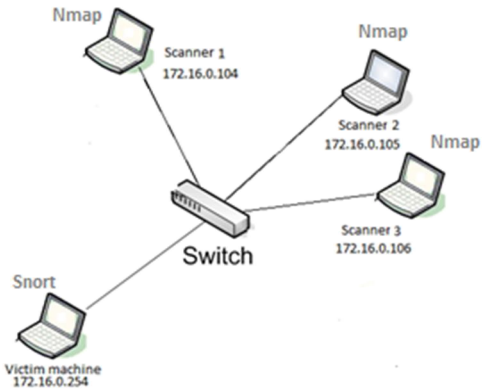


Fig. 10. Test bed Topology

In order to capture scanning traffic and performing relevant packet capturing an intrusion detection system, snort has been installed on victim machine.

Scanner 1, 2 & 3 performing port scanning simultaneously to the victim machine (in the presence of normal network traffic). A relevant packet capture run using the snort IDS and A significant reduction in log file is observed. Snort processed 15669(10.2 MB) and selects 50 (3.73 KB) packets and the improved decrement in the log size using relevant packet capture is about 99% of total traffic size. It is observed that, out of 19693 packets 50 is marked as relevant packets.

Machines which are responsible for scanning attempt are marked as suspicious scanner machines. Detection ability is 100%, due to the fact that the TCP packet without the SYN, ACK and RST bits set (Any combination of the other three FIN, PSH, and URG are OK) never appear in the normal TCP operation.

Table 1: Test-bed machines configuration summary

Machine Name and IP Address	**Operating system, Hardware configuration**	**Applications**	**Application Configuration/ Command**
Victim machine 172.16.0.254	Ubuntu 12.04 LTS. Intel dual core 2.50 GHz 1GB RAM	Snort 2.9.5.3	NIDS mode with Fast alert Snort rules: Protocol: TCP Rule Option: flags- !SRA
Scanner 1 172.16.0.104	Microsoft Win. 7 professional Intel dual 2 Duo 2.40 GHz 1 GB RAM	nmap 6.47(an windows version of Nmap)	**Nmap –sF – p** 20,21,22,23,25,53,67,68,69,143 **172.16.0.254**
Scanner 2 172.16.0.105			**Nmap –sX –P** 22,25,67,68,69,80,514,1812,2049,3306 **172.16.0.254**
Scanner 3 172.16.0.106			**Nmap –sFN** 20,21,23,53,443 **172.16.0.254**

Table 2: Identified and Scrutinize Suspicious Activity

Suspicious IP address	Port Observed	Date	Start Time	End Time	Scanning Variant
172.16.0.104	20,21,22,23,25,53,67, 68,69,143	27/11/2014	11:50:58	11:51:08	FIN
172.16.0.104	22,25,67,68,69,80,514 ,1812,2049,3306	27/11/2014	11:50:54	11:50:58	XMUS
172.16.0.104	20,21,23,53,443	27/11/2014	11:50:55	11:51:00	NULL

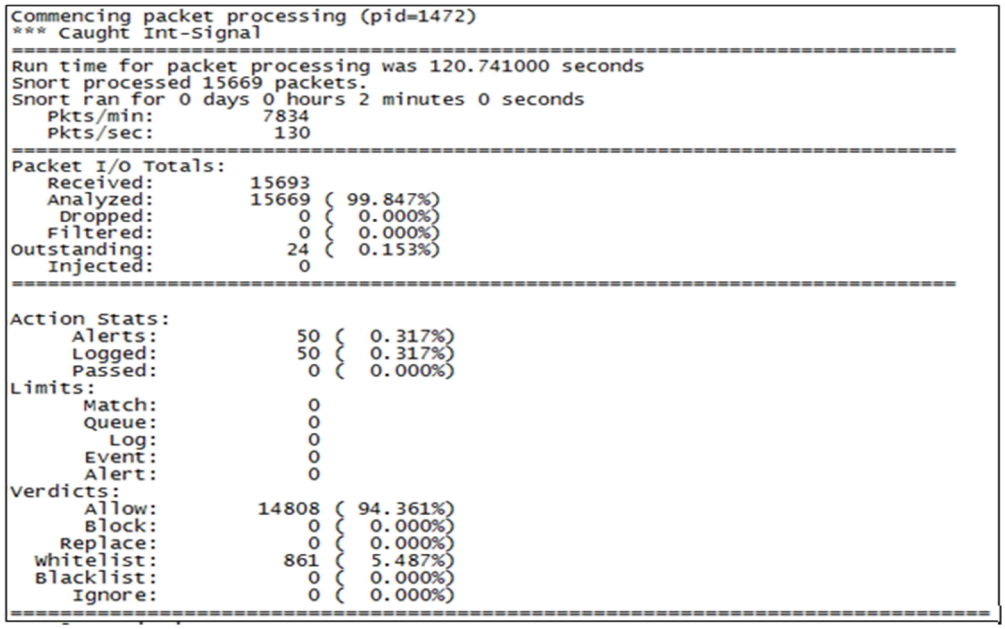


Fig. 11. Snort Packet processing Summary

9 CONCLUSION

In this work investigation of stealth port scanning attack is carried out on the basis of forensic principles. Proposed work presented a storage efficient capturing system that captures relevant packets and an analysis system that perform precise classification of suspicious packets. Snort rules are developed for the capturing and analysis of network traffic.

Experiment shows that, due to relevant packet capturing less than 1% of traffic data is logged and analysis performs 100% detection of suspicious packets.

This work extended in the future by including the detection and analysis of highly distributed, composite and hybrid, stealth port scanning techniques.

7 REFERENCES

[1] S. Panjwani, S. Tan, K.M. Jarrin, and M. Cukier. An experimental evaluation to determine if port scans are precursors to an attack. In Proc. Int. Conf. Dependable Systems and Networks, 2005. DSN 2005., pages 602 – 611, june-1 july 2005.

[2] R. Chandran, and S. Pakala,"Know Your Enemy: Learning about Security Threats", Addison Wesley, ISBN-10: 0321166469, 2004.

[3] Simson Garfinkel, Network Forensics: Tapping the Internet,http://www.oreillynet.com/pub/a/network/2002/04/26/nettap.html.

[4] Manowar H Bhuyan, D K Bhattacharyya and J K Kalita,"Surveying port scans and their Detection methodologies" The Computer Journal (2011) 54 (10): Pages 1565-1581 April 20, 2011.

[5] Whitepaper by dethy@synnergy.net "Examining port
[6] scan methods - Analyzing Audible Techniques" Published on 16 Oct. 2002 / Last Updated on 23 Jan. 2013, http://www.windowsecurity.com/whitepapers/misc/Examining_port_scan_methods__Analyzing_Audible_Techniques_.html.
[7] Brenden Claypool, Stealth port scanning methods, SANS Institute 2000 - 2002. As part of GIAC practical repository Claypool – 2002v1.4.http://www.giac.org/paper/gsec/1985/stealth-port-scanning-methods/103446
[8] Port Scanning Techniques, http://nmap.org/book/man- port-scanning-techniques.html
[9] [RFC793,http://www.rfceditor.org/rfc/rfc793.txt
[10] Snort, https://www.snort.org/
[11] MYSql,www. http://www.mysql.com/
[12] Nmap, www.nmap.org.