NETWORK SECURITY THROUGH SCRUBBER USING TCP/IP

R.Ragavendra[1], S.Kannadhasan[2]

[1,2] *Assistant Professor, Raja College of Engineering and Technology, Madurai, TamilNadu, (India)*

ABSTRACT

Transport scrubber or TCP/IP Scrubber is a transparent mechanism for explicitly removing network scans and attacks at various protocol layers. This proposed system utilizes the Winpcap 3.1 software which helps in establishing a connection between the kernel and the LAN. Packet capturing is mainly done by Jpcap Dumper incorporated in Winpcap 3.1. These packets are homogenized into TCP format by setting various flags. The percentage of packets captured in different protocols is displayed in run time. The transfer of packets between the end hosts are plotted from which the intruders is detected. The packet information of the intruder is collected and saved. The intruder's message is converted into a readable form by random assignment of values. Then the special characters in the message are scrubbed. The downfalls of firewalls and intrusion detection systems are overcome with the help of TCP/IP scrubber. It allows the detection of malicious contents, not only at the beginning, but throughout the flows life time. Thus, the ambiguities within the network are removed.

Keywords: Scrubbing. Sniffing, Tracing

I. INTRODUCTION

Firewalls primarily act as gate-keepers to a protected network. Due to performance reasons, when a firewall identifies an authorized flow, packets are routed through a fast path and are not scrutinized for further attacks. An intrusion detection system is a form of passive warden that observes network traffic in search of malicious attacks[1]. The ID system suffers a vantage point problem. It is vulnerable to attacks. As a solution to these problems TCP/IP scrubber is designed and implemented to achieve maximum throughput as well as a high level of security.

Two broad categories of intrusion detection systems: Network-based (NIDS) and Host based system. NIDS is implemented as passive network monitors that reconstruct networking flows and monitor protocol events through eaves-dropping techniques [2]. Host-based ID systems are located on the end-hosts in a network and monitor the resources and security procedures followed by co-resident users and applications [3]. Firewalls are effective means of protecting local system or network of systems from network-based security threats and provides access to outside world via WAN and the Internet. Firewall technologies are closely related to TCP/IP scrubber.

Both of them are active interposition mechanisms that is packets must physically travel through them in order to reach their destinations and both of them operate at the ingress points of a network.

II. TCP/IP SCRUBBER

The TCP scrubber is an active mechanism that explicitly removes ambiguities from external network flows, enabling downstream NIDS to correctly predict the end host response to these flows. The TCP scrubber's job is to codify what consists of well-behaved protocol behavior and to convert external network flows to this standard.

It allows detection of malicious contents, not only at the beginning, but throughout the flow's lifetime. It provides high performance as well as enforcement of flow invariants .

Two instances of scrubbers are:

1. Transport scrubber
2. Fingerprint scrubber

Transport scrubber supports downstream passive network-based intrusion detection systems by converting ambiguous network flows into well-behaved flows. Fingerprint scrubber restricts an attacker's ability to determine the operating system of a protected host. The design of TCP scrubber can be illustrated by comparing it with a full transport level proxy. A transport proxy is a user-level application that listens to a service port waiting for connections. When a client establishes a new connection an automatic connection is established between the proxy and the server. It just blindly reads and copies data from one connection to another. More costs is involved in this as TCP processing is done for both sets of connections in terms of throughput and scalability. Unlike the latter approach the TCP scrubber leaves the bulk of processing to the end points.

The two main tasks the TCP scrubber performs are:

- It maintains the current state of the connection
- Keeps a copy of the byte stream sent by the external host, but not acknowledged by the internal host It makes sure that the byte stream is always consistent and it throws away any packet which may lead to inconsistencies

The **Fig 1** represents the reduced TCP state processing that occurs at the TCP scrubber.

The three general states of the scrubber are :

- Connection establishment (INIT and INIT2)
- Established operation (ESTAB)
- Connection termination (CLOSE and CLOSED)

TCP scrubber does not keep any data state until the internal service host has acknowledged and reciprocated the TCP connection. It scales significantly because the amount of state kept in scrubber is much less than that kept at a transport proxy.. The TCP scrubber's approach to homogenization of TCP flows improves scalability in the number of simultaneous connections it can service.

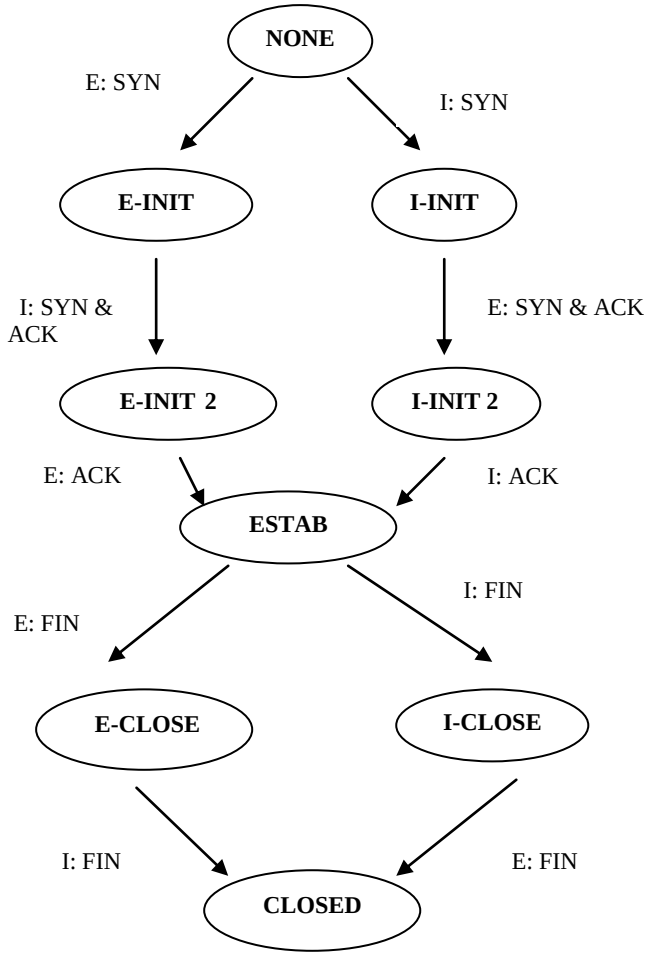

Fig 1: TCP state transition diagram

The working of a TCP scrubber can be explained with the help of an example as shown in the figures. In this example an attacker is trying to log into an end host as root trying to fool the NID system. The attacker can easily hack because the NID system and the end host reconstruct overlapping TCP sequences differently.

The figures given below represent the working of the TCP/IP scrubber towards an attack and how it solves the problem.

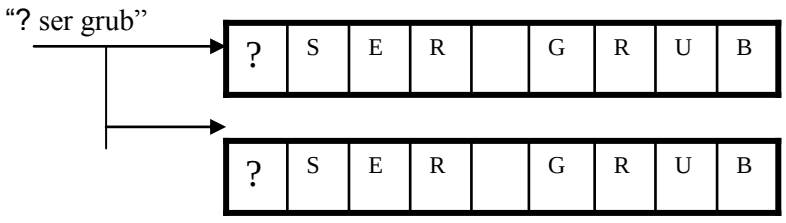

Fig 2: Host and NID system after attacker sends a hole

In fig 2, an attacker sends a data sequence to the end host with a hole at the beginning. Since TCP byte stream is reliable the end host and NID wait until the hole is filled before proceeding.

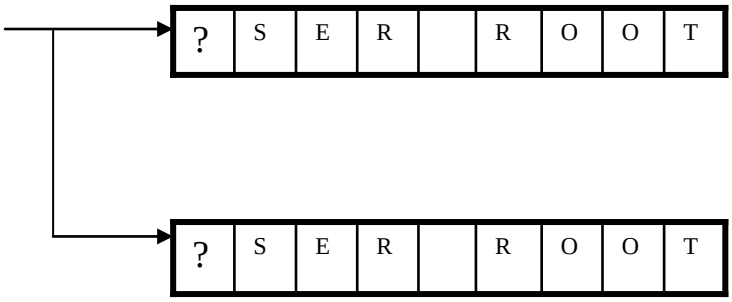

Fig3: Attacker filling in the hole and confusing the NID system

In fig 3, the attacker resends the data with the hole filled with a different username of the same length. Upon retransmission, a correct TCP implementation would always send the same data and it's up to the end host and the NID system as to which specification it should follow. The end host chooses to keep the new sequence of byte that comes in the second packet whereas the NID system keeps the first sequence of bytes.

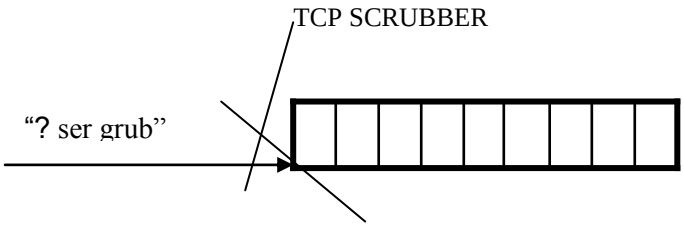

Fig 4: Scrubber enforces single interpretation

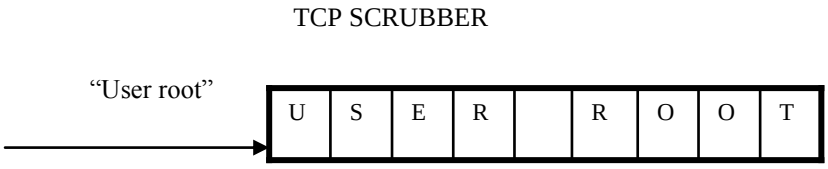

Fig.5: Attacker filling in the hole and sending new data

In Fig 4 and Fig.5, demonstrate how the active TCP scrubber interposed between the attacker and the downstream systems eliminates the ambiguity. The TCP scrubber enforces a single interpretation of the attackers TCP/IP stream to eliminate ambiguity. Here, the scrubber simply throws away the data after the hole, following which both the NID system and the end host see the attacker logging in as root.

III. SNIFFING OF PACKETS

Sniffers essentially capture, interpret and store for later analysis, packets traversing a network. The packet sniffer library set is 100% a fully self-contained, dynamically-loaded packet capture technology.

Packet sniffer may be used to develop:

- Hardware monitoring applications
- Applications, which use non standard network protocols
- Network traffic analyzers
- Network activity and load balance analyzers
- Traffic generators
- Network security scanners
- Network Intrusion Detection Systems etc

An Ethernet sniffer is software that works in concert with the network interface card (NIC) to blindly suck up all traffic within "earshot" of the listening system, rather than just the traffic addressed to the sniffing host. The sniffer software can capture and analyze any traffic that traverses the local Ethernet segment when the network hardware is in its promiscuous mode[4] .The capture session object represents a packet capture session. The methods in this object support enumeration of all network adaptors in addition to starting, pausing, resuming and stopping a packet capture from a single or from multiple network adaptors.

This packet filter object creates, enumerates, modifies, renames, deletes and applies a filter to a packet capture. The filter may be applied at capture time which means that only packets that meet the filter criteria will be saved, or it may be applied to an existing capture file.

The trace file object represents a packet trace file. Methods allow reading packets from an existing trace file as well as writing packets to a new or existing trace file. Existing trace files can be overwritten or appended to. Packets can be exported from one trace file object to another. This object need not be called during the actual capture. It is called when the resulting capture file is parsed.

This packet info object contains the packet details of retrieved packets and is populated by the trace file and capture session objects. It gives application access to the packet details such as packet length, the time the packet was captured, the actual packet content etc [5]. It is used along with the trace file and capture session objects to give packet details to the application. The protocol parser object is the interface to the connection of protocol parsers. It analyses the packet and breaks it up into the different protocol levels. Each protocol level includes two summary lines as well as multiple detail lines.

IV. IMPLEMENTATION

Packet capture is done with the help of packet sniffer. The sniffers allow the attackers to strike at every system that sends traffic to the compromised host, as well as any others sitting on the local network segment totally obvious to a spy in the midst.

Following are the steps performed during capturing:

- All packets of various protocols travel through the network.
- In addition, to the differences in their protocol specifications they also differ in their size, length, data content etc.
- Irrespective of the type of packets and differences in protocols all the packets are captured.
- After capturing the packet, it can be stored separately without passing it into the application layer or it can be destroyed without the users request.

The need for packet analyzer is that the user can exactly determine the nature of the packet. Analysis mainly takes place at four layers of the OSI model.

They are:

- Data link layer
- Transport layer
- Network layer
- Application layer

Following are the steps performed during analysis:

- The captured packets with encapsulated data and header information are analyzed from which the source and destination ports are obtained.
- A table is displayed demonstrating the source address, destination address, source MAC address and destination MAC address.
- A pie chart is also displayed in run time when the packets are captured. Using colors, the percentage of use of each protocol in different layers is displayed.

Intruders or hackers are people trying to get valuable information illegally. Hackers mainly perform hacking because:

- They are curious
- They are interested in breaking security
- Learning more about networks and systems

Following are the steps performed during the intimation of the intruder:

- A graph is plotted to show the various connections in the network.
- As soon as an intruder tries to snoop into our system, an intimation of the IP address of the attacker will be obtained in the graph with a different color.
- The encrypted form of the message is visible.
- The type of protocol used by the message is displayed along with the IP address.
- This immediate intimation helps in tracking down the intruder in the network.

V. EXPERIMENTATION AND RESULTS

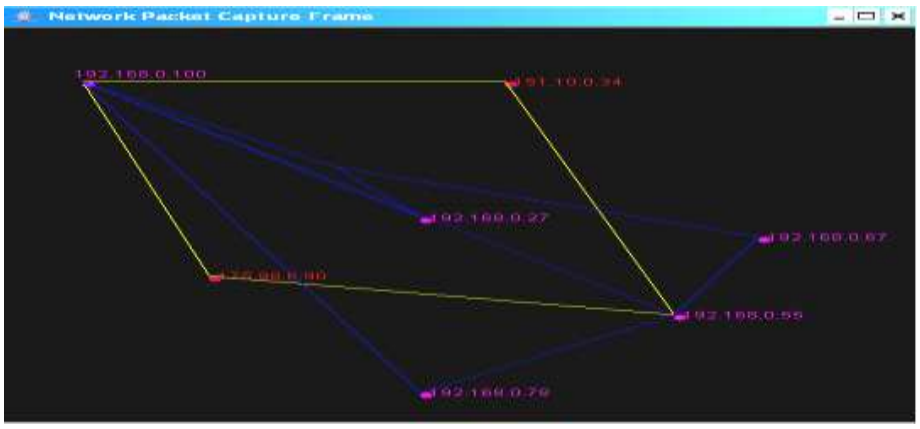

Fig 6: Plotting of the packets captured

Fig 6 plots the different packets captured along with the source and destination addresses. If an intruder tries to access this system without the permission, then his address is plotted with a different color to distinguish it from others. The authorized users are plotted in pink color and unauthorized users are plotted in red color.

The allowable IP and MAC address are shown below

IP ADDRESS	MAC ADDRESS
192.168.0.100	00-13-D4-3A-33-54
192.168.0.11	00-13-D4-3A-33-E9
192.168.0.78	00-17-9A-77-FE-30
192.168.0.27	00-17-9A-77-FE-9C
192.168..0.67	00-17-9A-77-FC-E7
192.168.0.55	00-17-9A-77-FC-E5
151.10.0.34	
175.18.6.90	

Table1: IP and MAC address

The IP addresses 192.168.0.100, 192.168.0.11 comprises of wired network The other IP addresses comprises of wireless networks.

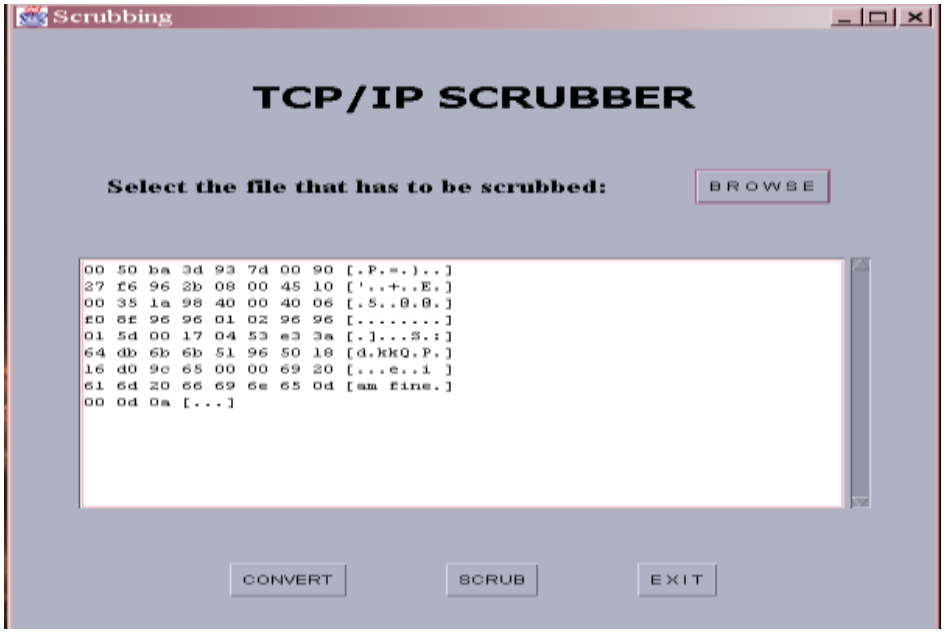

Fig 7: Intruder's Message

Based on the plotted figure, the information about the intruder is obtained with his IP address which can be viewed in Fig 7. The message of the intruder is collected and saved for further analysis.

The message that is received from the network is in an encrypted format that cannot be deciphered. This message has to be converted into a readable form as shown in Fig 8.

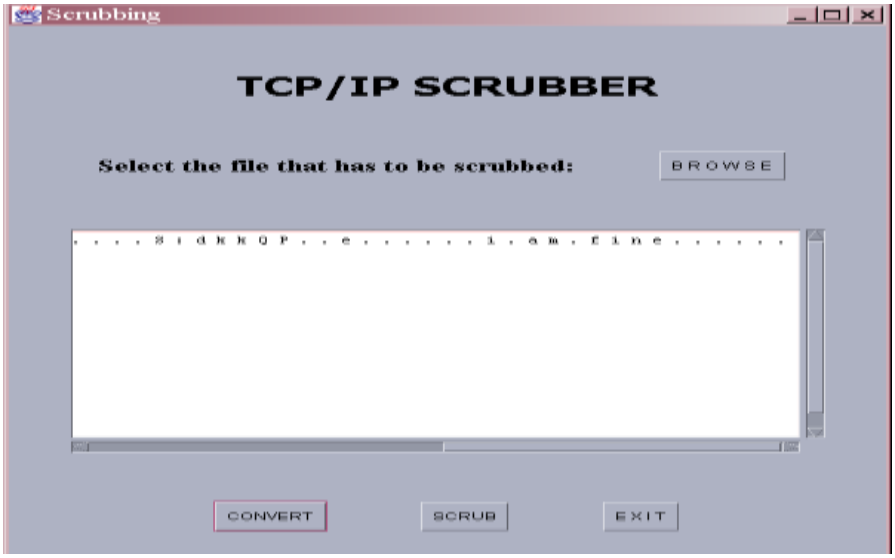

Fig 8: Conversion to readable form

VI. CONCLUSION

TCP scrubber is an active interposed mechanism for converting ambiguous network flows into well behaved flows that are interpreted identically at all downstream end points. When used in conjunction with NID systems the transport scrubber removes the attacks insuring a high confidence in the detection. It achieves high scalability and performance.

REFERENCES

[1] RFC 2460. Internet Protocol, Version 6 (IPv6) Specification.

[2] Cooper M, Yen DC. IPv6: business applications and implementation concerns. Computer Standards and Interfaces, vol. 28. Elsevier Science; 2005, 27–41.

[3] Zagar D, Vidakovic S. IPv6 Security: improvements and implementation aspects. In: Proceedings of the Eighth International Conference on Telecommunications, Contel. Zagreb; 2005.

[4] Sklavos N, Koufopavlou O. Mobile communications world: security implementations aspects – a state of the art. CSJM J Inst Math Comput Sci 2003;11(32):168–87, Number 2.

[5] RFC 4301. Security Architecture for the Internet Protocol.

[6] Molva R. Internet security architecture. Comput Networks, Vol. 31. Elsevier Science; 1999, 787–804.

[8] David Watson, Matthew Smart (2004) "Protocol Scrubbing: Network Security through Transparent Flow Modification", IEEE Transaction.

[9] Farnam Jahanian, G.Robert Malan (2004) "Transport and Application Protocol Scrubbing", IEEE Transaction.

[10] R.Atkinson, "Security Architecture for the Internet Protocol", RFC 1825.

[11] Karl N.Levitt, Todd Hererlein, "Network Intrusion Detection", IEEE Network, Vol.8.

[12] Addison-Wesley, W.Richard Stevens, "TCP/IP" Illustrated, and Volume 1: The Protocols. 6. Eric A.Fisch, Gregory B.White,"A Peer-Based Intrusion Detection System", IEEE Network, 10(1)