Network Forensic System for Port Scanning Attack

Atul Kant Kaushik, Emmanuel S. Pilli, R.C. Joshi
Department of Electronics & Computer Engineering,
Indian Institute of Technology Roorkee
Roorkee, India
{akk22pec, emshudec, rcjosfec}@iitr.ernet.in

Abstract—**Internet is facilitating numerous services while being the most commonly attacked environment. Hackers attack the vulnerabilities in the protocols used and there is a serious need to prevent, detect, mitigate and identify the source of the attacks. Network forensics involves monitoring network traffic and determining if the anomaly in the traffic indicates an attack. The network forensic techniques enable investigators to trace and prosecute the attackers. This paper proposes a simple architecture for network forensics to overcome the problem of handling large volumes of network data and the resource intensive processing required for analysis. It uses open source network security tools to collect and store the data. The system is tested against various port scanning attacks and the results obtained illustrate the effectiveness in its storage and processing capabilities. The model can be extended to add detection and investigation of various attacks.**

Keywords–network forensics, port scanning, NFATs, FIFO, snort

I. INTRODUCTION

Organizations are highly dependent on the internet for their businesses and incur heavy financial loss because of cybercrime. They are interested not only to detect and prevent these attacks but also to trace and prosecute the attacker. The field, attributing the attack, to a particular source, by capturing, analyzing and investigating the network traffic, is known as network forensics.

Network forensics includes the following basic processes: preparation, collection, preservation, examination, analysis, investigation and presentation. The collection and storage of the large volume of captured traffic data in an efficient way is challenging task. The processing and handling of this vast amount of data is also resource intensive. Much of this data collected may not be relevant for the analysis.

There are many open source and proprietary security tools like Wireshark [1], Tcpdump [2], and Snort [3] which can be used for network forensic analysis. These tools are used by network administrators to help in the collection of network traffic. They capture live network packets, analyze them, and present them in a format understandable to the user.

Network Forensic Analysis Tools (NFATs) allow administrators to monitor networks, gather all information about anomalous traffic, assist in network crime investigation and also help in generating a suitable response [4]. They can replay, isolate and analyze an attack or suspicious behavior and bolster network defenses accordingly. NetIntercept [5] and NetDetector [6, 7] are two proprietary hardware NFATs. PyFlag [8, 9] (Python Forensic Log Analysis GUI) and SiLK [10, 11] (System internet Level Knowledge) are two open source software NFATs. PyFlag handles network traffic captures logged in standard pcap format. SiLK works with packets captured in the Cisco NetFlow records.

The tools mentioned above are limited as they need to handle large amounts of data and a variety of formats must be analyzed to investigate the source of attack. These tools also require the forensic investigator to be an expert in network protocols and packet structure. This paper proposes a simple framework for network forensics which uses a popular intrusion detection system (IDS), Snort, to capture the network traffic. It uses FIFO for saving storage space, marks packets suspicious if having attack features and performs analysis to validate the attack. The implementation in C has been tested for port scan attack. Results giving the count of all the ports scanned and their port numbers, duration of scan and nature of attack are presented.

Section II gives the background and the general process framework involved in network forensics. Section III explains various port scan attacks and also identifies the features to mark the packets as suspicious. The architecture of the Network Forensic System (NFS) is explained in section IV. Section V explains the implementation. The results are discussed in section VI and the paper concludes with challenges and future work in section VII.

II. BACKGROUND

Network forensics is the field of applying forensic science to the computer networks in order to discover the source of network crimes. The objective is to identify malicious activities from the traffic logs, discover their details, and to assess the damage [12].

Network forensics is defined as "the use of scientifically proven techniques to collect, fuse, identify, examine, correlate, analyze, and document digital evidence from multiple, actively processing and transmitting digital sources for the purpose of uncovering facts related to the planned intent, or measured success of unauthorized activities meant to disrupt, corrupt, and or compromise system components as well as providing information to assist in response to or recovery from these activities" [13].

Network forensics is basically about monitoring, capturing, analyzing the network traffic and investigating the security policy violations. Forensic specialist monitors the network continuously and stores a copy of all or the relevant packets, depending upon the policy, in a prescribed format for future analysis. These stored packets' information is further analyzed, either manually or by using different approaches, to find if there is an anomaly in the network and if that anomaly is an attack. If any attack is found, the type of attack is determined and the source of the attack is investigated. Forensic specialists can attribute the attacker by proper monitoring, capturing, and analysis of the network traffic and by proper investigation.

A. *Classification of Network Forensics*

Network forensic systems are classified into two types each based on various characteristics like purpose, collection and nature:

- Purpose: 'General Network Forensics' to enhance network security & 'Strict Network Forensics' to get evidence satisfying the legal principles [14].
- Collection of Traffic: 'Catch-it-as-you-can' systems where all the packets passing through a particular traffic point are captured and analysis is subsequently done requiring large amounts of storage & 'Stop-look-and-listen' systems where each packet is analyzed in memory and certain information is saved for future analysis requiring a faster processor [15].
- Nature: The network forensic system may be an appliance with hardware and preinstalled software or exclusively software as in the case of PyFlag [8].

B. *Network Forensics Process Formalization*

The general process of network forensics is formalized into six steps: (i) Preparation and authorization (ii) Collection of network traces and logs (iii) Preservation and protection (iv) Examination and analysis (v) Investigation and attribution and (vi) Presentation and review. The network forensics specialists go through all these processes in order to find the network attacks and the source of the attack. The generic framework for network forensics is shown in Fig. 1. This model for network forensics was built as a special case of digital forensic models proposed in the literature [13, 14, 16, 17, 18].

Network forensics is applicable only to environments where network security tools (sensors) like intrusion detection systems, packet analyzers, and firewalls are deployed at various strategic points on the network. Preparation is needed to deploy these tools and configure them. The required authorizations to monitor the network traffic are obtained so that privacy of individuals and the organization is not violated. Any unauthorized events and anomalies noticed will be analyzed. The presence and nature of the attack is determined from various parameters.

Data is acquired from the sensors used to collect traffic data. The sensors used must be secure, fault tolerant, have limited access and must be able to avoid compromise. The integrity of data logged and network events recorded must be ensured. Collection is the most difficult part as traffic data changes at a rapid pace and it is not possible to generate the same trace at a later time. The amount of data logged will be enormous requiring huge memory space and system must be able to handle different formats appropriately.

The original data obtained in the form of traces and logs is stored on a back up device. A hash of all the trace data is taken and the data is protected. Another copy of the data will be used for analysis and the original collected network traffic is preserved. The traces obtained from various security sensors are integrated and fused to form one large dataset on which analysis can be performed. Redundant information is removed and minimum representative attributes are identified so that the least information with the highest probable evidence is analysis.

The collected data is classified and clustered into groups so that the volume of data to be stored may be reduced to manageable chunks. The evidence collected is searched methodically to extract specific indicators of the crime. The indicators are classified and correlated to deduce important observations using the existing attack patterns. Statistical and data mining approaches are used to search the data and match attack patterns.

The information obtained from the evidence traces is used to identify who, what, where, when, how and why of the incident. The observations are presented in an understandable language to the organizations management and legal personnel while providing explanation of the various standard procedures used to arrive at the conclusion. The results are documented to influence future investigations and in improvement of security products.

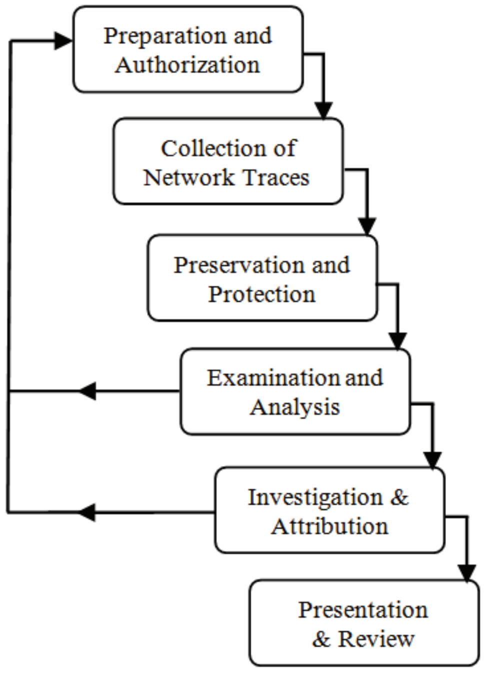
Preparation and Authorization
Collection of Network Traces
Preservation and Protection
Examination and Analysis
Investigation & Attribution
Presentation & Review

Figure 1. A Generic Framework for Network Forensics

III. Port Scanning

Port scanning is the process of identifying the listening ports of the victim system. It is one of the most popular techniques used to discover and map services that are listening on a specified port. Using this method an attacker creates a list of potential weaknesses and vulnerabilities in the open port leading to the exploitation and compromise of a remote host.

Many of the port scanning methods use the packet fields in the TCP protocol. The ports of a victim machine can be scanned in the following ways [19]:

A. SYN Scan

SYN scan is the most popular port scan for a number of reasons. It is very quick, scanning thousands of ports per second, on a fast network which is not hampered by restrictive firewalls. SYN scan is relatively stealthy and is referred to as half-open scanning because attacker doesn't make a complete TCP connection. The attacker checks for open ports by sending SYN packets in succession to different ports. Open ports respond with a SYN-ACK, and closed ports with a RST. The traffic logs will show a large number of SYNs and RSTs.

B. Connect Scan

The attacker attempts to make a full connection with each port to determine whether it is open by issuing the connect system call. A failed connection indicates the port is closed. This is a slow scan, and might also turn up in the system logs. The traffic log will show a large number of short-lived connections. This method not only takes longer and requires more packets to obtain the same information, but target machines are more likely to log the connection.

C. FIN Scan

The attacker sets the FIN flag on to bypass some firewalls that do not block FIN packets to identify whether a port is open. If the port is closed the machine will send a RST and if the port is open, it will ignore the FIN. The traffic logs will show a large number of FINs without corresponding SYNs and ACKs.

D. ACK Scan

This is used to find out which ports are filtered by a firewall. Some firewalls in the past did not block ACK packets, and this could be used to differentiate between filtered ports and closed ports. Both open and closed ports would respond with a RST, whereas filtered ports would give no response. This could be used to map the firewall rule sets of firewalls that are not stateful. The traffic logs will show a large number of ACKs without corresponding SYNs.

TCP SYN scan is the most commonly used method because of its simplicity and performance. Most of the scan tools like Advanced Port Scanner [20] and Free Port Scanner [21] use this method. We have used the above tools to launch the TCP SYN port scan attack on a 'victim' machine and have successfully gathered information about the attack, total number of ports scanned and port numbers scanned.

IV. Network Forensic System Architecture

The proposed Network Forensics System (NFS) which is designed to discover the source of network attacks is shown in Fig. 2.

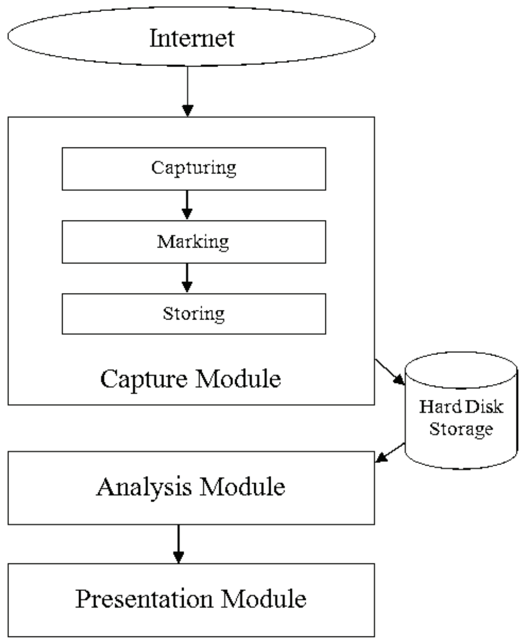

Figure 2. Network Forensic System Architecture

The proposed system architecture is a generalized architecture and can be extended to work for investigation of any type of network attack. We refer this architecture as Network Forensics System (NFS) to trace the source of port scanning attack. The system is designed to capture the network traffic from the network interface of the host, mark the relevant traffic packets, analyze the marked packet and display valuable information about all the suspicious IP addresses.

An IP address is classified as suspicious if it has tried to scan the ports of the host. The architecture of the system can be broadly classified into three modules: capture module, analysis module and presentation module. Each packet first passes through the capture module which marks it as either *relevant* or *irrelevant* based on a marking module which observes the flags in the packet. Only the packets marked as relevant pass to the analysis module which analyzes each incoming packet. It classifies each IP address that wants to connect to host as either *normal* or *suspicious* and finally provides valuable information about all the sources of port scanning attack.

This architecture described includes modules implemented for three of the six phases (ii, iv and vi) as explained in the generic framework. This framework will be extended to include the Investigation module which will trace the source of the attacker and attribute the attack by replaying the attack.

A. Capture Module

The main task of capture module is to capture the network traffic that is relevant for network forensic analysis. This module can further be subdivided to three sub modules: capturing, marking and storing.

1) Capturing: This module just captures all the packets passing through the network interface of the host and redirects it to a form of UNIX IPC (Inter Process Communication) named as FIFO (First-In-First-Out). The main advantage of using FIFO is that once the data has been read from it, data is discarded automatically from FIFO by UNIX kernel [22]. This procedure saves the storage space required to store the network traces.

This module uses snort to capture packets from network interface and redirects the output of snort to FIFO by using a function 'system ("snort –v > <fifopath>")'. The –r option of snort can also be used to capture traffic from tcpdump formatted file.

2) Marking: The task of this module is based on [23] and marks the packets that are relevant for further analysis. The strategy to mark the packet as relevant may vary while investigating the source for different kind of attacks. Our algorithm is limited to the port scanning method and will be able to trace the attacker only if attacker has used TCP-SYN, TCP-ACK, TCP-FIN methods. The steps are listed as follows:

a) Read the details of packet from FIFO into local buffers.

b) If the protocol used in the packet is not TCP then mark the packet as *irrelevant* and break, else go to step *c*.

c) If any of the flags S, F or R is not set then mark the packet as *irrelevant* and break, else go to step *d*.

d) If only S flag is set and the source IP address is the IP address of host machine then mark the packet as *irrelevant* and break, else go to step *e*.

e) Mark the packet as *relevant*.

f) Repeat the above steps for each packet captured from network interface of host machine.

S, R and F flags are checked in the third step to mark the packet corresponding to the method used for port scanning as TCP-SYN, TCP-ACK, and TCP-FIN method.

3) Storing: The task of this module is to store the marked packets into hard disk of the host system for further analysis. C functions like fprintf and fputs have been used for storing the marked packets. This module ensures that only the actual packets used for port scanning attack are stored. This results in less amount of storage required as compared to the normal process which captures all the packets passing through network interface and stores them on the host system.

B. Analysis Module

The analysis module analyzes the log file stored in the host system by the capture module, in order to discover the source of network attacks. A data structure named *ipliststruct* is designed with the following fields: IP address, unique id, port count, date, start time, end time, and 65536 flag fields one for each port number. A list is maintained corresponding to each machine with a certain IP address that has sent connection request to the host machine referred as *relevant-IP-address* and *relevant-machine* correspondingly. The attributes of *ipliststruct* designed to discover the source of port scanning attack are given as follows:

- IP address: The value of this field is the IP address of the relevant-machine.
- Unique id: A unique value of Uid is assigned to each relevant-machine and that value represents the current count value of the total yet encountered relevant-machines including the current relevant-machine.
- Port count: The value of Port count represents the total number of the port of the host machine in which the relevant-machine has yet sent the request.
- Date: The date when the relevant-machine has sent the first connection request after the program is run.
- Start time: The time when the relevant-machine sends its first connection request to the host machine.
- End time: The time when the relevant-machine has sent its last connection request to the host machine.
- Port flag [0] – Port flag [65535]: a flag corresponding to each port number of the host machine, its default value is 0. It is set to 1 if relevant-machine sends a connection request to corresponding port number of the host machine.

The algorithm for the analysis module is given below. When a new packet comes and read from FIFO, the following steps are followed:

a) If only S flag is set then go to step 2, otherwise break.

b) If source IP address is not the IP address of the host machine then go to step 3, otherwise break.

c) Update the fields of the data structure *ipliststruct* for the current IP address based on the decision whether the current IP address has been encountered or not.

d) Repeat the above steps for each packet read from FIFO.

C. Presentation Module

All the relevant machines are classified as either suspicious or normal based on its corresponding port count value. A threshold value is taken based on the observations and if the port count value is greater than or equal to the threshold for any relevant-machine, then that machine is categorized as suspicious and as normal otherwise. The result of the analysis module is displayed for validating whether the port scanning attack was carried out. The details like total number of packets processed, total number of relevant packets, IP address, port count, date, start time, end time and the value i (if Port flag[i] is set) is displayed as output for all the suspicious machines.

V. Implementation

The proposed network forensics system architecture is designed and implemented to trace port scanning attack. It is implemented in Fedora v9, a flavor of UNIX using C. The virtualization environment VMware Workstation [24] creates virtual machines by running multiple operating systems on a single physical machine. These virtual machines can be run simultaneously and can connect to the internet. VMware provides a distinct IP address for each host.

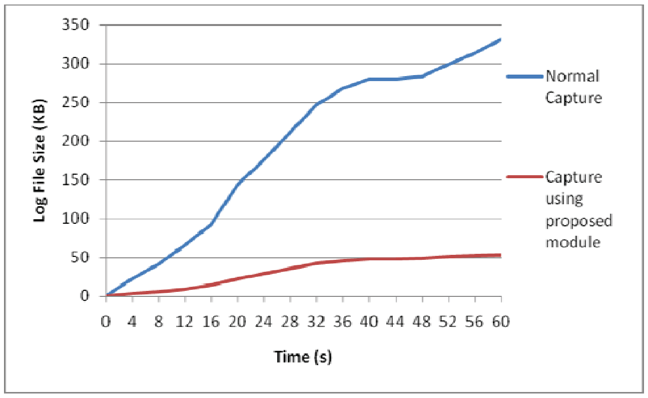

Figure 3. Comparison of normal capture with capture using proposed model

The popular intrusion detection system (IDS), Snort, is used for traffic capture and is installed in the Fedora v9, a guest operating system on the host Windows Vista in the VMware workstation. VMware provides two distinct IP addresses for host Windows Vista and guest Fedora v9.

Capture module is implemented separately according to the algorithm discussed in the above section. When this module is executed, it uses snort to capture all the traffic from the network interface. Snort is also run from the command line at the same time to capture the same traffic. The size of log files stored by both executions of snort is recorded after every four seconds.

The analysis module for port scanning is implemented slightly different from what is proposed in the architecture. This module is inserted between the marking and storing module of the capture module to effectively utilize the time as well as resource. When any packet is marked, it immediately enters the analysis module. The data structure *ipliststruct* is maintained based on the information from the analysis module. The implementation of the system to perform the two modules independently will need some more file operations and will perform slowly than the modified approach.

The system is implemented to choose either the real time packets or the packets from the log file of tcpdump format as the raw input packets. We have used log files as raw input as network forensics is usually performed *notitia criminis* (after notification of crime). VMware provided two distinct IP addresses for host Windows Vista and guest Fedora v9 facilitating port scan attack to be launched from Windows Vista host to Fedora v9 host. The program was run on Fedora v9. Two port scanning tools *Advanced Port Scanner* and *Free Port Scanner* were installed into the host operating system Windows Vista to launch port scanning attack.

VI. Results

The above system implementation has been executed and the results obtained are discussed here. A normal packet capture was run using the snort IDS and growth in file size with time was recorded. The capture module was executed and a significant reduction in the log file size was observed as shown in Fig. 3. The improved decrement in log file size after applying proposed capture module measured after 16, 32, and 48 seconds is 77.346 KB, 204.846 KB and 234.256 KB respectively.

The proposed model has been successful in addressing the issue of storage space for traffic logs. It also takes care of the processing capability as it is much feasible to analyze a small log file. The implementation has been tested for different port scanning attacks which were launched using *Advanced port scanner* and *Free port scanner* from the machine with IP address 192.168.127.1. The proposed system implementation worked well and gave the expected results which are given below.

A. *Result for scanned ports 6001 to 6010 using Advanced Port Scanner*

Port numbers starting from 6001 to 6010 of host system were scanned from Windows Vista having IP address 192.168.127.1 using *Advanced Port Scanner* and the traffic is captured using Snort then the traffics captured is passed to implemented system as input which gave the result:

Total number of processed packets	87
Total number of relevant packets	30
IP address	192.168.127.1
Total number of ports requested	10
Category	Suspicious
Date	09/21/2009
Start Time	01:38:10
End Time	01:38:19
Ports scanned	6001, 6002, 6003, 6004, 6005, 6006, 6007, 6008, 6009, 6010

B. *Result for scanned ports 21-23, 25, 53, 80, 110, 135, 137-139, 443, 445, 1080, 1433, 3128, 3306, 8080 using Free Port Scanner*

The ports listed above are scanned from the same system by using another port scanning tool *Free Port Scanner* then the system gave following result:

Total no. of processed packets	325
Total no. of relevant packets	55
IP address	192.168.127.1
Total number of ports requested to connect	18
Category	Suspicious
Date	09/21/2009
Start Time	14:28:33
End Time	14:34:19
Ports scanned	21, 22, 23, 25, 53, 80, 110, 135, 137, 138, 139, 443, 445, 1080, 1433, 3128, 3306, 8080

C. Result for normal request from 192.168.127.1

This is the case when one normal connection request came from the same system and it was categorized by the NFS as "normal". This is because it has not exceeded the threshold value set for the operating system Fedora v9.

Total no. of processed packets	30
Total no. of relevant packets	4
IP address	192.168.127.1
Total number of ports requested to connect	1
Category	Normal
Date	10/03/2009
Start Time	13:38:39
End Time	13:38:40
Ports scanned	80

VII. Conclusion and Future Work

A network forensics system is definitely a valuable investigation tool to discover the source of network attacks and to attribute them to the attacker. The proposed architecture of network forensics system is validated and proved to be useful by implementing it for one specific network attack called port scanning.

The result discussed above shows the resultant improvement in the storage space required to store the log file of captured packets by capturing only the packets those are relevant for the analysis of port scanning attack. It also shows that our implementation worked well in discovering the details of the attacker who scanned the ports of host machine using Advanced Port Scanner and Free Port Scanner.

In the future, we plan to extend the proposed architecture by including the fourth module, investigation module, which will trace the actual attacker if the IP discovered by the proposed architecture, is a spoofed one. Further, we also plan to implement the proposed system architecture to trace the source for various kinds of network attacks and plan to modify the current implementation to process the IPv6 packets.

References

[1] Wireshark, www.wireshark.org

[2] TCPDump, www.tcpdump.org

[3] Snort, www.snort.org

[4] R. Sira, "Network Forensics Analysis Tools: An Overview of an Emerging Technology," GSEC, Version 1.4, Jan., 2003.

[5] NetIntercept, "NetIntercept Datasheet," http://www.sandstorm.net/support/netintercept/downloads/ni40_datasheet.pdf, Last accessed October 25, 2009.

[6] NetDetector, www.niksun.com

[7] P. Venezia, "NetDetector captures intrusions," http://www.infoworld.com/d/security-central/netdetector-captures-intrusions-306, Last accessed October 25, 2009

[8] PyFlag, http://www.pyflag.net

[9] M.I. Cohen, "PyFlag - an advanced network forensic framework" *Digital Investigation (The International Journal of Digital Forensics & Incident Response)*, vol. 5, no.1, pp. 112-120.

[10] SiLK, http://tools.netsa.cert.org/silk

[11] Gates, M. Collins, M. Duggan, A. Kompanek and M. Thomas, "More Netflow Tools: For Performance and Security," in *18th Conference on Large Installation Systems Administration*, Atlanta, USA, 2004, pp. 121-132.

[12] R. Chnadran, "Network Forensics", in Know Your Enemy: Learning about Security Threats, Ed. L. Spitzner, Second Edition, Addison Wesley Professional, 2004, pp 281 - 325.

[13] G. Palmer, "A Road Map for Digital Forensic Research," *1st Digital Forensic Research Workshop (DFRWS 2001),* pp. 27–30, 2001.

[14] W. Ren and H. Jin, "Modeling the network forensics behaviors," *Proc. 1st Int'l Conf. Security and Privacy for Emerging Areas in Communication Networks* (SecureComm 2005), Sept., 2005, pp. 1–8

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

[16] M. Reith, C. Carr, and G. Gunsch, "An examination of digital forensic models," *International Journal of Digital Evidence,* vol. 1, pp. 1-12, 2002.

[17] B. Carrier and E. H. Spafford, "Getting physical with the digital investigation process," *International Journal of Digital Evidence,* vol. 2, no. 2, pp. 1-20, 2003.

[18] V. Baryamureeba and F. Tushabe, "The enhanced digital investigation process model." *Proc. 4th Digital Forensic Research Workshop* (DFRWS 2004).

[19] J. Gadge and A.A. Patil, "Port Scan Detection," Proc. *16th IEEE International Conference on Networking* (ICON 2008), New Delhi, Dec., 2008.

[20] Advanced Port Scanner v1.3, http://www.radmin.com/products/utilities/portscanner.php Last accessed September 25, 2009.

[21] NSAsoft LLC, Free Port Scanner v2.8, http://www.nsauditor.com/network_tools/free_port_scanner.html Last accessed September 25, 2009.

[22] W.R. Stevens, "Pipes and FIFOs", in Network Programming, Vol II, Second Edition, PHI, 1995, pp 61-92.

[23] A. Almulhem and I. Traore, "Experience with Engineering a Network Forensics System" Proc. International Conference on Information Networking (ICOIN 2005), pp 62-71.

[24] VMWare Workstation, http://www.vmware.com/products/workstation/ Last accessed September 25, 2009