Slow Port Scanning Detection

Mehiar Dabbagh[1], Ali J. Ghandour[1], Kassem Fawaz[1], Wassim El Hajj[2], Hazem Hajj[1]
[1] Department of Electrical and Computer Engineering
[2] Department of Computer Science
American University of Beirut
E-mails: {mmd43, ajg04, kmf04, we07, hh63}@aub.edu.lb

Abstract- Port scanning is the most popular reconnaissance technique attackers use to discover services they can break into. Port scanning detection has received a lot of attention by researchers. However a slow port scan attack can deceive most of the existing Intrusion Detection Systems (IDS). In this paper, we present a new, simple, and efficient method for detecting slow port scans. Our proposed method is mainly composed of two phases: (1) a feature collection phase that analyzes network traffic and extracts the features needed to classify a certain IP as malicious or not. (2) A classification phase that divides the IPs, based on the collected features, into three groups: normal IPs, suspicious IPs and scanner IPs. The IPs our approach classify as suspicious are kept for the next (K) time windows for further examination to decide whether they represent scanners or legitimate users. Hence, this approach is different than the traditional approach used by IDSs that classifies IPs as either legitimate or scanners, and thus producing a high number of false positives and false negatives. A small Local Area Network was put together to test our proposed method. The experiments show the effectiveness of our proposed method in correctly identifying malicious scanners when both normal and slow port scan were performed using the three most common TCP port scanning techniques. Moreover, our method detects malicious scanners that are otherwise not detected using well known IDSs such as Snort.

Keywords- Intrusion Detection System, Port Scanning.

I. INTRODUCTION

As a result of the large usage of computers and computer networks all over the world, computer network security has become an international priority. A significant challenge for today's network engineers and security administrators is to develop techniques capable of detecting attempts to compromise the integrity and availability of the network. This area of research is called Intrusion Detection Systems (IDS). Port scanning is regarded as a dangerous network intrusion method for discovering exploitable communication channels [1]. It is considered as the most popular reconnaissance technique attackers use to discover services they can break into [2]. Port scanning consists of probing a network host for open ports. Port scanning is a more targeted form of information gathering that attempts to profile the services that are run on a potential target.

Currently, different techniques are used to accomplish port scanning probes such as, TCP connect scanning, TCP half-connect scanning, Xmas Tree scanning and NULL scanning [3]. Port scanning is divided generally into two main parts, horizontal and vertical. In horizontal scans, the same port is scanned over multiple hosts. This is useful for attackers who want to gain control on victim hosts by exploiting a known vulnerability of a certain service running on that port. While in vertical attacks, multiple ports are scanned on the same host. This is common for attackers who are gathering information to attack a specific target host. In the context of this research, the focus will be on the vertical port scans.

Two basic approaches are used to detect intrusions: misuse detection and anomaly detection. Misuse detection attempts to recognize attacks that follow a certain intrusion pattern. Such patterns are collected from known attacks and are stored in the form of signatures in the database. Anomaly detection, on the other hand, can be identified by recording unusual behavior of operations. An anomaly is a feature that is out of the ordinary, e.g., abnormal network traffic which is actually caused by unknown attacks. An anomaly detection system models normal behavior and identifies a behavior as abnormal if it is sufficiently different from known normal behaviors [4][5].

In this paper, we propose a simple and efficient approach for detecting slow port scanning. The idea behind our detection method is monitoring traffic every time window, then classifying the IPs into scanner IPs, suspicious IPs and legitimate users based on extracted features from the traffic. The behavior of suspicious IPs is monitored in the following time windows in order to decide whether they represent scanners or legitimate users, which allows the detection of slow port scanning.

The rest of the paper is organized as follows. In section II, we overview the methods designed to detect port scanning and we describe the techniques used by attackers to make port scanning stealthier and difficult to detect. We also highlight in this section how our method is different from the traditional detection approaches. In section III, we explain our proposed method for detecting these scans. Experiments and results are shown in section IV. In section V, we conclude the paper and present our future work.

II. RELATED WORK

Recently, researchers have proposed various techniques to detect port scans. The authors in [6] outline several

978-1-4577-2155-7/11/$26.00 ©2011 IEEE

approaches to detect intrusions, including port scanning. More specifically, the authors propose techniques that correspond to both the misuse detection and anomaly detection. In [7], the authors used the number of the different TCP control packets as input for Back Propagation algorithm in order to detect port scans. The learning phase was based on a training set that contains normal traffic and port scanning attacks. In [8], the authors proposed detecting scanning attacks based on the number of ICMP error messages that are generated when the scanner tries to connect to a closed port. Hence, no algorithm was used for classifying the IPs. Alternatively, an attack is flagged when the number of ICMP error messages exceeds a predefined threshold. Fuzzy logic was used in [9] for detecting non distributed port scans. Under the category of misuse detection, the authors propose a two-stage rule induction algorithm, called the PNrule. In the first phase, the algorithm learns the P-rules that cover most of the intrusive examples while in the second, it discovers N-rules used to eliminate the false positives. Another method called CREDOS was suggested; it uses the ripple down rules to overfit the training data at the beginning and then prune them to improve the generalization capability.

Under anomaly detection category, the authors in [6] proposed several techniques namely, the Local Outlier Factor (LOF), the Mahalanobis-distance Based Outlier Detection, and Nearest Neighbor (NN) Approach. The output of these algorithms is a decision making device that relies on features processed from a time window or a connection window to classify scanners from normal users [10]. In [11] the authors proposed an anomaly detection technique based on k-means clustering algorithm in order to distinguish an attack from normal traffic. A close approach suggested in [12] considers large amount of data rather than a windowed data. It utilizes a relational database management system (MySQL) to perform OLAP like operations (group by) using SQL statements. However, this approach scales poorly to the increase in the request rate at the server, and will incur delays that might largely exceed network delays.

In [13], the authors define in details how port scanning problem can be a text-book data mining problem. By so they define features, data transformation for the gathered traffic, data labeling procedure, and the choice of the classifier (named Rapper). The authors in [14] assign an anomaly score to a source IP based on the number of failed connection attempts it has made. It operates under the assumption that port scanners will induce more failed connections. However, this approach's performance relies greatly on the chosen thresholds and the definition of the failed connection. The research in [15] uses likelihood based detection to detect whether a connection is normal or represents a scan. However, since the access is skewed towards normal traffic (99% of the time), the algorithm results in high percentage of false positives. Another anomaly detection system based on traffic analysis, SPICE [16], utilizes entropy like function to see whether the accessed port is probable or not, with lower probability inducing more information. The algorithm sums the negative log-likelihood of destination IP/Port pairs until it reaches a given threshold. Nevertheless, a single scan on a single port can result in a false positive. Finally, the current state-of-the-art for scan detection is Threshold Random Walk (TRW) proposed in [17]. It traces the source's connection history performing sequential hypothesis testing. The hypothesis testing is continued until enough evidence is gathered to declare the source either scanner or normal.

Techniques used for port scanning are well known and last normally for a short time period, the fact that allowed popular IDSs design methods that effectively detect port scanning. The traditional and most common approach to detect port scanning is the time-based detection method. This method is based on monitoring traffic in every time window. The IDS decides at the end of each time window if there is a scanning activity based on the behavior of the IPs in that time window. Snort is a very well-known and widely used IDS that uses that approach in detecting port scanning [18]. Snort detects a port scan if the number of connections-to-closed ports in a time window x exceeds a certain number of connections y, where x and y are determined by the user and x is usually in minutes. This is based on the fact that legitimate users know the available services and shouldn't make too many connections to closed ports. However, an attacker willing to spend more time to scan a specific server can launch a slow port scanning. The attacker may try to connect to one port every half an hour and the IDS will not detect the scan.

Another approach used to detect slow port scanning is the connection-based detection method. This methods was suggested in [6] and it relies on labeling network connections. Connection-based features are the same as time-based features that are used in the time-based detection method, but they are computed for the last N connections, i.e. they are connection windowed rather than time windowed. This connection window is mainly utilized to cope with the slow port scans. However, similar to the attack used to deceive the time-based features, the attacker may choose to generate legitimate connections greater than the window size N between each successive port scans.

To detect slow port scanning attacks, the IDS would be required to capture the traffic for a larger period of time. Such an approach to detect slow port scanning is proposed in [12]. However, it can be shown that this approach incurs a large delay on the server side and therefore the client will witness a huge degradation in the Quality of Service (QoS). Furthermore, an IDS operating on a large time window might become a target for a Denial of Service (DoS) attack by overloading the server with too many traffic that requires a lot of processing.

It is worth noting that distributed port scanning is another technique that is used by attackers to harden the detection of port scanning. A distributed port scanning is defined as a many-to-one host scanning where different ports on the destination host are being scanned by multiple

attackers [2]. In a distributed slow port scanning, the number of ports being scanned by a single attacker is relatively small, thus being able to fool the IDS.

In this paper, we propose a simple and efficient method for detecting slow port scanning. Our proposed method monitors traffic in small time windows. The reason behind choosing small time windows is to keep the amount of traffic processed small. Consequently, our method does not cause degradation in services for clients and protects the server from becoming a victim of a DoS attack However, our method remains capable of detecting slow scans by classifying IPs in each time window based on their behavior into three categories: scanner IPs, suspicious IPs and legitimate users. Our IDS examines the behavior of the suspicious IPs in the following time windows and decides whether they are scanners or legitimate users. This is different from the traditional IDSs (like Snort), where IPs are classified into either scanners or legitimate users in each time window solely. This technique allows attackers to perform undetected scans by choosing a large interval between the probes. Our proposed method detects the three most common TCP port scanning techniques: the connect scan, the half-connect scan and the FIN scan.

We used thresholds in classifying IPs rather than using a machine learning algorithm. This is due to the fact that a machine learning algorithm requires a training data so that the algorithm can learn how to classify the IPs. Finding a representative training data for slow port scanning is difficult. A machine learning algorithm that is trained by a training data that has collected traffic for a slow scan where the interval between the scanning probes is 3 minutes might not classify the IPs correctly if the interval between the probes is made 6 minutes by the scanner. A solution to overcome this problem will be to use an adaptive algorithm that changes its classification rules over time. However, an attacker who has some knowledge on the used algorithm can deceive the IDS by making the adaptive algorithm learn the scanning activity as normal causing high false positive and negative alarms. The details of our method are discussed next.

III. PROPOSED METHOD

Our intrusion detection system focuses on TCP scans, which are the most common used techniques. The method is based on collecting features for every IP in a predefined time window. In order to detect slow port scans, we need a large time window so that we can detect any abnormal pattern in the behavior of the clients. However, choosing a large time window (one hour for example) will require a large memory for storing the traffic and will cause a large delay at the server side as there will be a large amount of traffic that needs to be checked for many IPs; the fact that will lead to performance degradation. Alternatively, we choose a small time window (T) where (T) is in minutes. Our IDS collects different features for each IP address every (T) minutes. The IP addresses are then classified based on the collected features that reflect their behavior, into 3 groups: normal IPs, suspicious IPs and scanner IPs. In order to allow the detection of any abnormal behavior in a time that is larger than the time window (T), we keep the suspicious IP addresses in a list for the following (K) time windows and we call that list the suspicious IPs list. This will allow the IDS to watch the behavior of these IPs in a larger period. If any IP that is a member of the suspicious IPs list is again classified as suspicious IP, the module will notify the administrator or firewall about this abnormal behavior.

In order to define the collected features, we need some knowledge on the way a TCP connection is established and terminated in normal cases. We also explain how the *connect*, the *half-connect* and the *FIN* scans are performed and show how our collected features distinguish scanners from legitimate users.

A TCP connection is established by a three-way hand shake as shown in Figure 1. A client who wants to connect to a specific port sends a TCP segment with the SYN bit in the header of the segment set on. The server on the other side receives this segment and checks if there is an available service on that port then it responds by a TCP segment with the SYN and ACK bits set on. Finally, the client completes the three-way handshake and replies by an acknowledgment. Otherwise (if there is no available service for that port), the server responds to the received SYN segment by a segment with the RST bit set on (i.e. RST bit has a value of 1) [19].

To terminate an established TCP connection, the client sends a TCP segment with the FIN bit set on. The server responds by sending another TCP segment with the FIN and ACK bits set on. Finally the client acknowledges the reception of that segment by sending a segment with the ACK bit set on as shown in Figure 2.

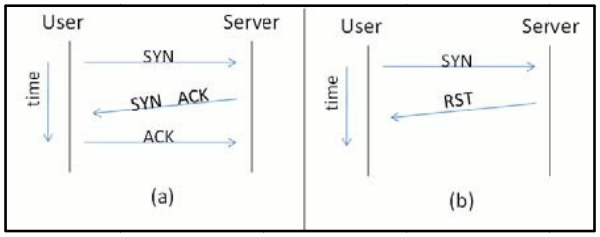

Fig. 1 TCP connection establishment: (a)- the three-way handshake for establishing a connection between the user and an open port. (b)- A connection attempt between the user and a closed port.

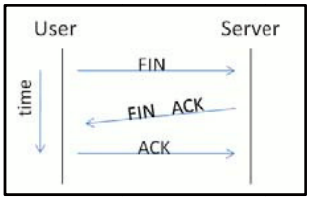

Fig. 2 Terminating an established connection.

The three most common TCP scanning techniques that an attacker may follow in order to know which ports are open and which ports are closed are: connect scan, half-connect scan and FIN scan. We next explain how these scans are performed and for each scan, we present the features that will be collected in order to distinguish legitimate users from scanners.

a) Connect Scan: the scanner sends a SYN, and determines whether the port is open or closed based on the server's response. If the server replies by a SYN-ACK then the port is open and the scanner continues the three-way hand shake and establishes a connection by sending an ACK to the server. Otherwise, the server replies by a RST to declare a closed port.

The first feature that distinguishes legitimate users from scanners is the number of connections made to closed ports (N_{closed}). Legitimate users are aware of the offered services and are not expected to try to establish connections to closed ports. On the contrary, there is a great chance that an attacker while scanning the ports will try, unknowingly, to connect to closed ports. For each IP, the number of connections attempted to closed ports (N_{closed}) is determined by calculating the difference between the incoming SYN from each IP and the outgoing SYN ACK from the server to each IP. The number of connections to closed ports that are made by an IP (ip) can be determined as shown in equation (1):

$$N_{closed}^{ip} == SYN_{in,ip} - SYN_{out,ip} \quad (1)$$

Where $SYN_{in,ip}$ is the number of incoming SYN segments from (ip) to the server, and $SYN_{out,ip}$is the number of outgoing SYN-ACKS that are sent from the server to (ip).

If no connections are made to closed ports, then the difference is equal to zero. This is based on the fact that when the server receives a connection establishment to a closed port, it responds by an RST indicating that the destination port is closed. Whereas the server responds by a SYN ACK segment if the port is open.

b) Half-Connect Scan: This technique is also known by the SYN scan. It is similar to the connect scan except that after receiving a reply from the server that the port is open, the scanner doesn't continue the three-way hand shake and doesn't establish a full connection. This type of scan is usually more difficult to detect, since establishing the connection is not completed and isn't logged.

The number of half connections (N_{hc}^{ip}) that are made by an IP (ip) can be determined directly by calculating the number of SYN-ACKs that were sent from the server to (ip) and that weren't followed by an ACK.

c) FIN Scan: As discussed before, the FIN bit is set when the two parties want to terminate a pre-established connection. A scanner determines whether a port is open or closed by sending a TCP segment with the FIN bit set to a port. Since no connection to that port was established in the first place, the server responds by RST if the port is closed or the server doesn't respond at all if the port is open. Based on whether the scanner receives an RST or not, he can determine whether the port is closed or open.

The number of FIN probes that were not preceded by establishing a connection (N_{Fin}) can be calculated for each IP by calculating the difference between the incoming FINs from the IP to the server and the outgoing FINs from the server to the IP. The number of FIN probes that are made by an IP (ip) and that weren't preceded by establishing a connection can be determined as shown in equation (2):

$$N_{Fin}^{ip} = FIN_{in,ip} - FIN_{out,ip} \quad (2)$$

$FIN_{in,ip}$ is the number of incoming FINs that are sent from (ip) to the server, and $FIN_{out,ip}$ is the number of outgoing FINs from the server to (ip).

The difference should be zero in normal cases since the connection is terminated by exchanging two FIN segments (one coming from the host and one coming from the server) as explained before.

We calculate these features ($N_{closed}, N_{hc}, N_{Fin}$) for each IP at the end of each time window. Then we decide whether each IP represents a scanner or a legitimate user based on these features. The rules for classifying an IP (ip) based on the values of its features are shown as follows:

$$state(ip) = \begin{cases} legitimate, N_{closed}^{ip} = 0 \; and \; N_{hc}^{ip} = 0 \; and \; N_{FIN}^{ip} = 0 \\ suspicious, \;\; N_{closed}^{ip} = 1 \; or \; N_{hc}^{ip} = 1 \; or \; N_{FIN}^{ip} = 1 \\ scanner, N_{closed}^{ip} > 1 \; or \; N_{hc}^{ip} > 1 \; or \; N_{FIN}^{ip} > 1 \end{cases}$$

If all the collected features are equal to zero, then the IP is classified as a legitimate user since all the IP's activity in that time window is normal. If N_{closed}^{ip} or N_{hc}^{ip}or $N_{Fin}^{ip}$is equal to one, then we can't decide whether (ip) is a legitimate user or a scanner. If $N_{closed}^{ip}$is equal to one, then it could be a legitimate user who made by mistake a connection to a closed port or a scanner who is performing a stealthy slow scan. Also if N_{hc}^{ip} equals one, then it is possible that the legitimate user tried to connect to an open port by sending a SYN segment, but the legitimate user's machine was shutdown, for some reason, before completing the three-way hand shake causing a half-open connection. A normal case where N_{Fin}^{ip}equals one is when the legitimate user wants to terminate an established connection by sending a FIN segment to the server. But in case of network congestions, the server won't reply by a FIN segment directly. The legitimate user will send another FIN segment assuming that the first one was lost. Once the first FIN segment is received by the server, it replies with a FIN segment and terminates the connection, whereas the server ignores the second received FIN segment and the difference between the incoming FINs and outgoing FINs will equal one. All of the above mentioned cases occur rarely, but should be taken into account in order to reduce false positive alarms.

So what we do is add this IP to the suspicious list to further examine its behavior. If the IP isn't classified as suspicious in the following (K) time windows, it is removed from the suspicious list. Otherwise, an alarm is sent to the administrator since this continuous suspicious behavior represents a slow port scanning. The suspicious list is a table that contains IP addresses; the table has a small size and is the only thing that is saved after the time window expires. If N_{closed}^{ip} or N_{hc}^{ip}or $N_{Fin}^{ip}$is larger than one, then (ip) is classified as a scanner and an alarm is sent to the administrator or a message is sent directly to the firewall to disable incoming connections from that IP.

The duration of the time window (T) and the number of consecutive time windows (K) after which the suspicious IP is removed from the suspicious list are specified by the user. If T = 3 minutes and K = 10, then the only way to launch an undetected slow port scan will be to scan one port every (T×K= 30 minutes). This is due to the fact that our detection method will classify the IP as suspicious, but after 10 consecutive time windows it will be removed from the suspicious list since it might be a legitimate user who incorrectly made a connection to a closed port. However, if the scanner is willing to spend 30 minutes to scan each port, this requires a very long time for scanning all the 65535 TCP ports to know the available services. Traditional IDS systems need to process the collected traffic every 30 minutes in order to achieve results similar to our detection which requires a lot of processing for the collected traffic since the number of packets exchanged in 30 minutes is huge. Also these traditional IDS systems won't detect a scan if the interval between the probes is 30 minutes. This proves that our method provides a more efficient solution compared to the traditional techniques.

IV. EXPERIMENT AND RESULTS

We implemented our proposed method and tested it on a small local area network. A testbed was built comprised of a victim machine, a Layer 2 switch, and four other machines as shown in Figure 3. Three of the machines are scanners and the fourth is a legitimate user accessing services provided by the victim machine. We assume that the victim machine provides services and by such is a server.

To test our detection technique, a java code was developed to perform the three types of scans (the connect scan, the half-connect scan and the FIN scan), where the time interval between the sent probes can be set by the scanner. The code was run on the three scanner machines and the time interval between the scanning probes was set to be 0.4 seconds, 4 minutes and 6 minutes for scanner1, scanner2 and scanner3 respectively. These intervals were chosen to represent both normal port scanning (scanner1) and slow port scanning (scanner 2 and 3). In fact, a very popular scanning tool called Nmap uses an interval between the probes equal to 0.4 seconds as a default value when normal port scanning is performed. The legitimate user establishes normal TCP connections to the server (victim machine), exchanges data and then terminates the connection.

To implement our detection method, we developed a Java code that captures traffic at the network interface of the victim server using the JPCap library. The time window (T) is set to be 3 minutes, so that based on the traffic of each IP in three minutes, we calculate the three features ($N_{closed}, N_{Half\_connect}, N_{Fin}$) and we divide the IPs based on these features into scanner IPs, suspicious IPs and legitimate users. A suspicious IP is removed from the suspicious list after (K=10) consecutive time windows.

Our detection method was able to identify that scanner 1 is performing port scanning in the first time window and a notification was sent to the administrator of detecting port scanning coming from the IP of the scanner1. Scanners 2 and 3 who were performing slow port scans (where the interval is equal to 5 and 6 minutes) were classified as suspicious IPs in the first time window and were added to the suspicious list. By observing the behavior of the scanners in the following 10 time windows, our detection method notices that their behavior is again abnormal. Actually, our approach detected the malicious scanning behavior in the second and third time windows for Scanner 2 and Scanner 3 respectively and alarmed the administrator of detecting slow port scanning. The legitimate user wasn't misclassified as a scanner. Our experiment proves that our method is able to correctly detect both normal and slow port scanning without giving false alarms. Our detection method implemented in java can be found in [20].

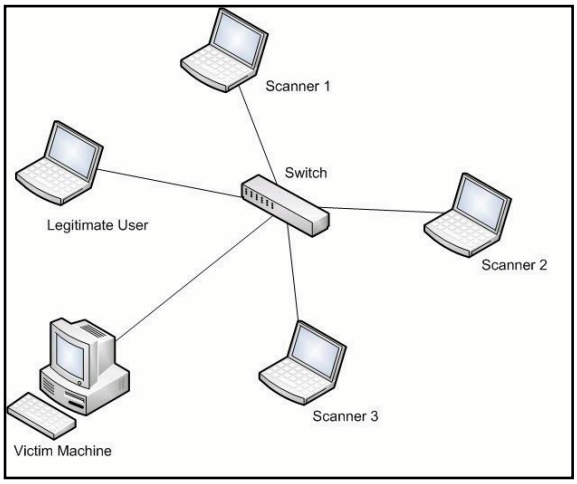

Fig. 3 Testbed Topology, the victim machine represents a server that offers services, the legitimate user establishes normal TCP connections with the victim server, one of the three scanner is performing normal port scanning and the other two are performing slow port scanning on the victim server.

V. CONCLUSION AND FUTURE WORK

In this paper we propose a simple and efficient approach for detecting slow port scanning. Our method processes the captured traffic in a small time window and therefore overcomes the disadvantages of the previous approaches that work on a large time window, thus requiring a lot of processing which causes degradation in the QoS and might become a target for a DoS attack. Our approach divides the IPs into three categories: scanner IPs, suspicious IPs and legitimate user which is different than the traditional IDS that classify the IPs into either scanners or legitimate users. These traditional IDS can't detect slow port scanning. On the other hand, our method blocks scanner IPs directly while suspicious IPs are monitored in the upcoming time windows and either cleared or blocked. The experimental results show the effectiveness of our approach in correctly classifying IPs and in correctly detecting normal and slow port scanning. For future work, we plan to implement our method on larger traffic data and determine the probability of false detection for our method. We also plan to propose solutions for the distributed port scanning which is the second stealthy technique after slow port scanning that is used by attackers to hide the scanning activity.

REFERENCES

[1] U. Kanlayasiri, S. Sanguanpong and W. Jaratmanachot, "A Rule-based Approach for Port Scanning Detection", *23nd Electrical Engineering Conference (EECON 23)*, Chiangmai November, 2000.

[2] W. El-Hajj, H. M. Hajj, Z. Trabelsi, and F. Aloul, "Updating Snort with a Customized Controller to Thwart Port Scanning", *Security and Communication Networks Journal*, 2010.

[3] G. Lyon, "Port Scanning Techniques", Internet: http://nmap.org/book/man-port-scanning-techniques.html, Jan.1, 2009 [Mar. 18, 2010].

[4] D. Kang, "Learning Classifiers for Misuse and Anomaly Detection Using a Bag of System Calls Representation", *In Proceedings of the 6th IEEE Workshop on Information Assurance and Security United States Military Academy*, West Point, NY,2005.

[5] W. Lee and S. J. Stolfo, "Data mining approaches for intrusion detection", *In Proceedings of the 1998 USENIX Security Symposium*, 1998.

[6] P. Dokas, L. Ertoz, V. Kumar, A. Lazarevic, J. Srivastava and P. Tan, "Data mining for network intrusion detection", *In Proc. 2002 NSF Wrokshop on Data Mining*, p. 21-30.

[7] B. Soniya and M. Wiscy, "Detection of TCP SYN Scanning Using Packet Counts and Neural Network," *Signal Image Technology and Internet Based Systems, 2008. SITIS '08. IEEE International Conference*, pp.646-649, Nov. 30 2008-Dec. 3 2008 doi: 10.1109/SITIS.2008.33

[8] H.U. Baig and F. Kamran, "Detection of Port and Network Scan Using Time Independent Feature Set," *Intelligence and Security Informatics, 2007 IEEE* , pp.180-184, 23-24 May 2007 doi: 10.1109/ISI.2007.379554

[9] J. Kim and J. Lee , "A slow port scan attack detection mechanism based on fuzzy logic and a stepwise policy," *Intelligent Environments, 2008 IET 4th International Conference*, pp.1-5, 21-22 July 2008

[10] L. Ertoz, E. Eilertson, A. Lazarevic, P. N. Tan, P. Dokas, V. Kumar and J. Srivastava, "Detection of novel network attacks using data mining," *In Proc. of Workshop on Data Mining for Computer Security*, 2003.

[11] H. Yang, F. Xie and Y. Lu; , "Research on Network anomaly Detection Based on Clustering and Classifier," *Computational Intelligence and Security, 2006 International Conference* , pp.592-597, Nov. 2006 doi: 10.1109/ICCIAS.2006.294204

[12] S. Jahr, "Slow portscanning detection," Internet: http://www.ztian.org/docs/slow_portscanning_detection.pdf, Nov. 2005 [Mar. 22, 2010].

[13] G. J. Simon, H. Xiong, E. Eilertson and V. Kumar, "Scan detection: A data mining approach," *In Proceedings of the Sixth SIAM International Conference on Data Mining*, 2006, pp. 118–129.

[14] P. A. Porras and A. Valdes, "Live traffic analysis of TCP/IP gateways," in NDSS, 1998,

[15] C. Leckie and R. Kotagiri, "A probabilistic approach to detecting network scans," *In Proceedings of the Eighth IEEE Network Operations and Management Symposium (NOMS 2002)*, 2002, pp. 359-372.

[16] S. Staniford, J. A. Hoagland and J. M. McAlerney, "Practical automated detection of stealthy portscans," *Journal of Computer Security*, vol. 10, pp. 105-136, 2002.

[17] J. Jung, V. Paxson, A. Berger and H. Balakrishnan, "Fast portscan detection using sequential hypothesis testing," *In Proceedings of 2004 IEEE Symposium on Security and Privacy*, 2004, pp. 211-225.

[18] http://www.snort.org.

[19] A. S. Tanenbaum, "Computer Networks, Fourth Edition", 2003.

[20] http://www.rprojects.webs.com .