

3.1 Introduction:

In this chapter, 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 port scans and slow port scans by classifying IPs in each time window based on their behaviour into two categories: scanner IPs, suspicious IPs. Our IDS examines the behaviour of the suspicious IPs in the following time windows and decides whether they are scanners or not. 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 probe s 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. [1]

3.2 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 behaviour 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 behaviour, into 3 groups: normal IPs, suspicious IPs and scanner IPs.

In order to allow the detection of any abnormal behaviour 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 behaviour of this 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 behaviour.

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 3.1**. 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).[31]

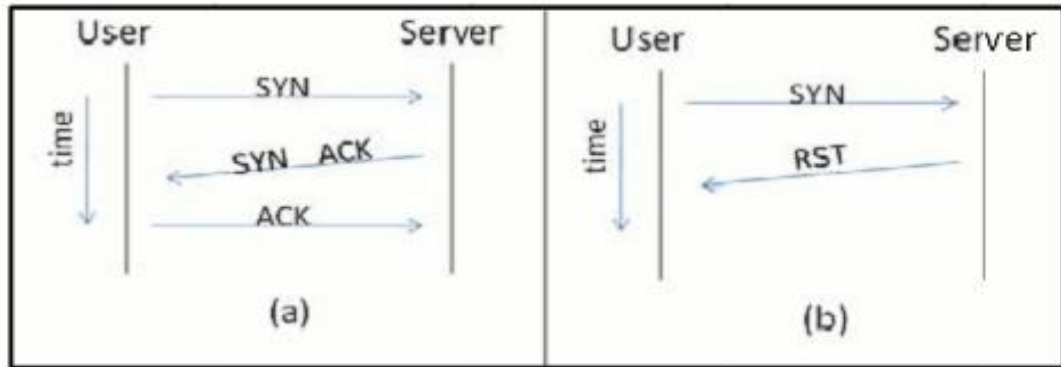


Figure 3.1 TCP connection establishment: (a) - the three-way handshake for establishing a connection between the user and an open port. (b)- A connection attempt between the user and a closed port.

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 3.2**.

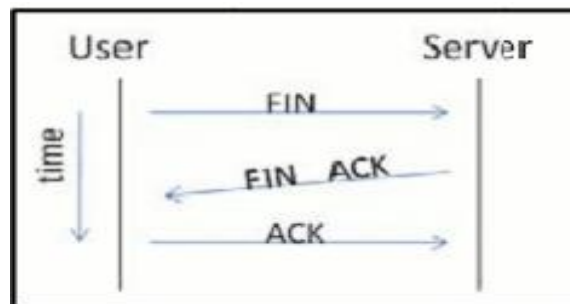


Figure 3.2 terminating an established connection.

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 as shown in equation (2) :

$$N_{hc}^{ip} = SYN-ACK_{out,ip} - ACK_{in,ip} \quad (2)$$

- 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 (3):

$$N_{Fin}^{ip} = Fin_{in,ip} - Fin_{out,ip} \quad (3)$$

$Fin_{in,ip}$ 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 (4):

$$state(ip) = \begin{cases} \text{legitimate}, & N_{closed}^{ip} = 0 \text{ and } N_{hc}^{ip} = 0 \text{ and } N_{Fin}^{ip} = 0 \\ \text{suspicious}, & N_{closed}^{ip} = 1 \text{ and } N_{hc}^{ip} = 1 \text{ and } N_{Fin}^{ip} = 1 \\ \text{scanner}, & N_{closed}^{ip} > 1 \text{ and } N_{hc}^{ip} > 1 \text{ and } N_{Fin}^{ip} > 1 \end{cases} \quad (4)$$

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 with its destination port to the suspicious list to further examine its behaviour. 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 behaviour represents a slow port scanning. The suspicious list is a table that contains IP addresses and their destination port; the table has a small size and is the only thing that is saved after the time window expires.

If the IP is classified as suspicious in the following time windows, we test its destination port, which is our contribution in this work.

The value of the destination port can be classified in three classes:

- If the current destination port is the successor of the destination port of the same ip address in the suspicious list then (ip) is classified as a scanner.
- If the current and the previous destination port of the same ip address in the suspicious list in the set of known services port (80, 53, 25, 21....) then (ip) is classified as a scanner.
- If the current destination port has any value and its IP address is repeated 3 times in the suspicious list then (ip) is classified as a scanner.

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.

3.3 Algorithm To Detect Slow Port Scan:

```

read (k)
WT:= readWindowTime();

while (true) do{
i :=0 ;
    while i<k {
        ST:=readSystemTime();
        while (ST<ST+WT) do {
            receivePacket(p);
            analyse (p);
            save(p);
            ST:=readSystemTime();
        }
        calculateConnect() {Nclosed = SYNIN, IP -SYN-ACKOUT, IP, ( SYN-ACKout, ip =ACKin, ip }
        calculate HalfConnect(){NHS = SYN-ACKOUT, IP -ACKin, ip }
        calculateFin();{ Nfin = FINin, IP - FINout, IP }
        if (Nclosed =1 or NHS =1 or Nfin =1) {
            if IP in (suspicious-list) {
                dst-port1:=extract( l.IP.dst-Port) ; nbr:= extract (l.IP.nbr)
                if ip.dst-port=dst-port1) or ip.dst-port in (services-port) or (nbr=k-1){
                    put (ip, scanner-list)
                }
                else l. IP.nbr=l. IP.nbr+1;}
            else (// IP not in suspicious list)
                IP.nbr=1;
                put ( IP , suspicious-list);}
        else (// not(Nclosed =1 or NHS =1 or Nfin =1) )
            if ( Nclosed >1 or NHS >1 or Nfin >1 )
                { put (ip , scanner-list)}
            }
        i:=i+1;}
empty (suspicious list);
}

```

3.4 Discussion:

Our method can quickly detect any attempt of scan in the second captured window (6 minutes), because the most scanners do a scan for successive ports, or a scan for the most known service port.

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 $K = 5$, then the only way to launch an undetected slow port scan will be to scan one port every ($T \times K = 15$ minutes). This is due to the fact that our detection method will classify the IP as suspicious, but after 5 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 15 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 15 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 15 minutes is huge. Also these traditional IDS systems won't detect a scan if the interval between the probes is 15 minutes. This proves that our method provides a more efficient solution compared to the traditional techniques.

3.5 Conclusion:

In this chapter we have presented a detailed description of our detector algorithm. Our system captures network traffic in a window and classifies into three classes (legitimate, suspicious and scanner), our detector after k times can determine whether the IP address in the suspicious list is a scanner or not. The full implementation of this work is in the next chapter.