

1 Introduction

Security is a big issue for all networks in today's enterprise environment. Hackers and intruders have made many successful attempts to bring down high-profile company networks and web services. Many methods have been developed to secure the network infrastructure and communication over the Internet, among them the use of firewalls, encryption, and virtual private networks. Intrusion detection is a relatively new addition to such techniques. Intrusion detection methods started appearing in the last few years. By using intrusion detection methods, you can collect and use information from known types of attacks and find out if someone is trying to attack your network or particular hosts. The information collected this way can be used to harden your network security, as well as for legal purposes.

Port scanning is the most famous technique used by hackers in recent years because it is easy to scan, but it is difficult to detect Port scanning. In the following, we will deal with the different existing types of scans and more specifically the TCP scan.

2 Intrusion

An intrusion is a deliberate move into someone else's territory either literal or figurative. If someone breaks into your home, that is also an intrusion. First used in the late 14th century, the noun intrusion derives from the Latin word intruder, which combines the prefix in-, meaning "in," and truder, meaning "to thrust, push." If someone reads your diary, that is considered an intrusion of privacy. Ordering a Muslim woman to take off her veil would be considered an intrusion in religious beliefs. You may remember intrusion used in science class to describe molten rock that forms in an earlier rock formation. [36]

3 Intrusion detection

Intrusion detection is a set of techniques and methods that are used to detect suspicious activity both at the network and host level. Intrusion detection systems fall into two basic categories: signature-based intrusion detection systems and anomaly detection systems. Intruders have signatures, like computer viruses, that can be detected by using a software. You try to find data packets that contain any known intrusion-related signatures or anomalies related to Internet protocols. Based upon a set of signatures and rules, the detection system is able to find and log suspicious activity and generate alerts. Anomaly-based intrusion detection usually depends on packet anomalies present in protocol header parts.

In some cases, these methods produce better results compared to signature-based IDS. Usually, an intrusion detection system captures data from the network and applies its rules to that data or detects anomalies in it. [22]

4 Definition of intrusion detection system (IDS)

Intrusion detection (ID) is defined as “the problem of identifying individuals who are using a computer system without authorization (i.e., ‘crackers’) and those who have legitimate access to the system but are abusing their privileges (i.e., the ‘insider threat’).” For our work, we add to this definition the identification of attempts to use a computer system without authorization or to abuse existing privileges. Thus, our definition matches the one given in [18], where an intrusion is defined as “any set of actions that attempt to compromise the integrity, confidentiality, or availability of a resource.” [9]

5 Types of Intrusion Detection System (IDS)

There are two main types of Intrusion Detection System (IDS):

-  Signature Based IDS (SBIDS)
-  Anomaly Based IDS (ABIDS). [17]

In SBIDS, also known as misuse detection, signatures of known attacks are stored and the events are matched against the stored signatures. It will signal an intrusion, if a match is found. The main drawback with this method is that it cannot detect new attacks whose signatures are unknown. This means that an IDS using misuse detection will only detect known attacks or attacks that are similar enough to a known attack to match its signature. [17]

ABIDS has attracted many academic researchers due to its potential for addressing novel attacks. Novelty detection is the identification of new or unknown data that a machine learning system is not aware of during training [17]. ABIDS have two major advantages over signature based intrusion detection systems. The first advantage is the ability to detect unknown attacks as well as “zero day” attacks. This is because of the ability of anomaly detection systems to model the normal operation of a system/network and detect deviations from them. A second advantage is that the aforementioned profiles of normal activity are customized for every system, application and/or network, and therefore making it very difficult for an attacker to know with certainty what activities it can carry out without getting detected [17].

6 Classification of IDS

Intrusion Detection systems can be classified into three categories based on the types of data they examine. These are: [6]

-  Host Based IDS.
-  Network Based IDS.
-  Application Based IDS.

6.1 Host-based Systems

Host-based intrusion detection systems are aimed at collecting information about activity on a particular single system, or host. These host-based agents, which are sometimes referred to as sensors, would typically be installed on a machine that is deemed to be susceptible to possible attacks. The term “host” refers to an individual computer, thus a separate sensor would be needed for every machine. Sensors work by collecting data about events taking place on the system being monitored. This data is recorded by operating system mechanisms called audit trails. Other sources from which a host-based sensor can obtain data, “include system logs, other logs generated by operating system processes, and contents of objects not reflected in standard operating system audit and logging mechanisms.” These logs are for the most part simple text files, which are written a few lines at a time, as events occur and operations on a system take place. [28]

6.2 Network-based Systems

Network-based intrusion detection systems offer a different approach. “These systems collect information from the network itself,” rather than from each separate host. They operate essentially based on a “wiretapping concept.” Information is collected from the network traffic stream, as data travels on the network segment. The intrusion detection system checks for attacks or irregular behavior by inspecting the contents and header information of all the packets moving across the network. The network sensors come equipped with “attack signatures” that are rules on what will constitute an attack, and most network-based systems allow advanced users to define their own signatures. This offers a way to customize the sensors based on an individual network’s needs and types of usage. The sensors then, compare these signatures to the traffic that they capture. This method is also known as packet sniffing , and allows the sensor to identify hostile traffic. [28]

6.3 Application Based IDS

Application-Based Intrusion Detection Systems (IDS)

a. Description

Application-Based IDS is a special subset of Host-Based IDS (HIDS) that analyzes the events transpiring within a software application. The most common information source for Application-Based IDS is the application's transaction log file. [27]

b. Rationale

The ability to interface with applications directly allows Application-Based IDS to detect suspicious behavior such as users exceeding their security authorization. [27]

c. Benefits

Application-Based IDS monitors the interaction between user and application, which traces activity to individual users.

Application-Based IDS works with applications that access encrypted data since it interfaces with the application at transaction endpoints where information is presented to users in unencrypted form. [27]

7 IDS Architecture

Basically, IDS has two types of architecture; the first one is centralized and the second one is distributed. In a centralized architecture, all activities such as monitoring, gathering, analyzing, detecting, reporting, and so forth, are all done by a central location. This central location will do all of the mentioned tasks for the whole network all by itself. Distributed architecture is when multiple IDS are used to monitor large network where monitoring and detection is being controlled by a local control node, which will report to one or multiple central locations in hierarchical manner. All multiple IDS communicate with each other. Centralized architecture is vulnerable to attack because if the central location is down, then, the entire network is open to attackers. Distributed architecture is more reliable because multiple IDSs are used so that if one IDS is down there are backups, due to this reason distributed environment is used for this study. [9]

8 Limitations of existing IDS

Many of the existing network- and host-based IDSs [9] perform data collection and analysis centrally using a monolithic architecture. We mean that the data is collected by a single host, either from audit trails or by monitoring packets in a network, and analyzed by a

single module using different techniques. Other IDSs [9] perform distributed data collection (and some preprocessing) by using modules distributed in the hosts that are being monitored. But the collected data is still shipped to a central location where it is analyzed by a monolithic engine. A good review of systems that take both approaches is presented in [9].

There are a number of problems with these architectures:

- ✚ The central analyzer is a single point of failure. If an intruder can somehow prevent it from working, the whole network is without protection.
- ✚ Scalability is limited. Processing all the information at a single host implies a limit on the size of the network that can be monitored. After that limit, the central analyzer becomes unable to keep up with the flow of information. Distributed data collection can also cause problems with excessive data traffic in the network.
- ✚ It is difficult to reconfigure or add capabilities to the IDS. Changes and additions are usually done by editing a configuration file, adding an entry to a table or installing a new module.
- ✚ Analysis of network data can be flawed. As shown in [9], performing collection of network data in a host other than the one to which the data is destined can provide the attacker with the possibility of performing Insertion and Evasion attacks.[9]

In a centralized architecture, there exists a single point of failure, that is, if the IDS fails due to attacks, breakdowns or other reasons, the whole network is vulnerable to attacks because attackers could sneak into the network undetected as mentioned by Mohamad Eid (2005): “A single machine monitors data flow at a strategic point in the network and collects and analyzes data from the log files. Once an attacker destabilizes this host, he or she is able to gain considerable access to the whole network.” [11]

9 Useful Characteristics of MAs

MAs have many characteristics that enable them to enhance intrusion detection technology. Mobility is obviously one of the most important capabilities, and we can certainly benefit from it. However, other agent capabilities also lend themselves to intrusion detection technology. Agent technology and agent applications, mimic collections of autonomous and intelligent individuals. Classes of individuals have special purposes and each can operate independently from the others. Each individual talks to other individuals that it meets and exchanges information. This paradigm is in sharp contrast to the traditional programming paradigm where a master logic unit controls a set of slave units. The slaves have no autonomy and perform exactly what the master logic unit commands. Variations of the traditional

approach include multiple units with set duties and set communication channels. There may not be a central controller and each unit may rely on the other units to perform their job. If one unit ceases to function, the other units are not intelligent enough (or do not have the authority) to solve the problem. This traditional distributed programming paradigm works well when components can be relied upon to function. Even by using redundant components, an attacker can disable a small finite number of backups. This traditional design is easy to implement and is an efficient solution to many problems. Agent technology is a great contrast to this design since it attempts to give each agent an understanding of its environment along with the authority to independently make decisions. [26]

MAs are by nature autonomous, collaborative, self-organizing, and mobile. These features are not found in traditional distributed programs, and enable IDSs to implement completely new approaches for doing intrusion detection, some of which are based on analogies found in nature and in society. [26]

Picture a collection of MAs as a colony of bees. Each bee has the ability to fly to flowers and glean pollen just like MAs can move among hosts and process data. Bees have no need to carry large flowers home. Similarly, MAs can avoid having to transfer intrusion detection data located at a collection point, back to a central repository. This research is described under Multi-Point Detection. [26]

Another analogy is to picture a collection of MAs doing IDS work as a colony of ants. Step on one or even hundreds of the workers, and the colony continues to function. In addition, ants can move away when they see a foot descending. MAs can be built with similar attack resistance since they are autonomous and mobile, and killing a few should not harm an ideally constructed MA application. [26]

MAs can also be viewed as a collection of guards. Security companies do not want to incur the expense of posting a guard in every hall of a building. Instead, they have a guard walk through each hall periodically checking for intrusions. Likewise, MAs enable one to periodically check hosts for security problems without having to install checking software on every host. This research is described under Roaming Agents. [26]

These are just a few examples of how colonies of autonomous mobile agents can benefit intrusion detection technology. Mobility is an important aspect, but that alone is not sufficient. MAs need to be able to operate autonomously and operate in consort with other agents. These features enable new intrusion detection paradigms. [26]

10 Advantages and Disadvantages of Mobile Agents in Intrusion Detection

Several advantages related to mobile agent usage are listed in literature (Jansen et al. 1999, Lange and Oshima 1998, Smith 1988). Some of them are listed here: Delay caused by Networks: When hierarchical IDSs are used in a network, it results in slower response when an attack occurs. This is because; the central controller (machine) has to send the information about the attack and the response to be taken to that particular host through the network. This may not always result in immediate response against the attack, as the time taken for the information to reach the destination host might be too long. Thus, traditional hierarchical IDS may not be successful in achieving on time the detection of attacks. On the other hand, if mobile agents are used, they can respond faster as they are directly dispatched from the central controller to the target host. [13]

10.1 Minimizing the network traffic

Traditional IDS employed different data collection mechanisms to collect data both at the host and the network level. This data was later used to track any intrusions. Generally, the collected data is very huge and for an intrusion to be detected, data from different hosts have to be collected and processed by the central controller. This results in increasing the network traffic and creating an overhead on the network. By employing mobile agents, the load on the network can be reduced as these mobile agents employ efficient search mechanisms there by reducing the necessity for data traffic among several hosts. [13]

10.2 Persistency

As mobile nodes operate autonomously and asynchronously, they are not prone to failure even if the machine, which hosted them, fails. This provides an additional advantage of employing mobile agents in IDS. In the case of the centralized machines, when the central controller fails, the entire IDS is considered to be down as there is no communication among other hosts. [13]

10.3 Structure and Platform Independence

Mobile agents can be used in IDS with a flexible structure. For example, one agent can be designated for collecting the data in the network, the other agent can be used to detect and report anomalies while the rest of them can be used to take appropriate action. Due to this structure, the mobile agents find tremendous application in IDS. Also, mobile agents from

different vendors can be used to build IDS. Moreover, it is possible to write our own mobile code to make it applicable to the existing environment. [13]

10.4 Dynamic Nature

The dynamic nature of mobile agents enables them to be moved around the network. This makes it possible to reconfigure the system during runtime as well. Mobile agents can be cloned, dispatched or put to sleep when the network configuration has to be changed. Also, they can sense their execution environment and dynamically adapt to the situation. [13]

10.5 Heterogeneous Environment

Mobile agents can be interoperable on multiple platforms. This is possible because of the virtual interpreter installed on the host machine. Mobile agents are generally computer and transport-layer independent and are dependent only on the execution environment. This feature enables the mobile agents to be used on several different platforms without compatibility problems. [13]

10.6 Robust in Nature

Even if one of the agents fails, the other agents in the IDS can take up the tasks of the failed agent and continue the detection. This robust behavior of mobile agents makes them more applicable in large environments where several agents and their interaction is needed for proper monitoring of the network. [13]

10.7 Scalability

By employing distributed mobile-agent IDS, however large the network grows, it could be easily handled. Agents have the capability to clone and distribute themselves to the new machines when they are added to the network. [13]

- ***Drawbacks in using mobile agents***

✚ The main problem in using mobile agents rests in security. Mobile agents require administration rights as they initiate a response when an intrusion is identified. By granting a mobile agent all permissions to the host it is operating on, an intruder can easily induce any virus. These security problems are the main hindrances for vast growth of mobile technology. Some preliminary measures can be taken in order to alleviate these security problems. Some of them include providing limited access control to important resources, applying cryptographic methods to exchange information.

✚ One more potential problem involved, is when the mobile agent contains credit card details of the user. Some hosts might try to get the private information from the mobile agents, which contain client details. [13]

✚ Observing the manner in which, the network attacks are increasing. It becomes obligatory on IDS to detect attacks immediately and report them spontaneously. If mobile agents are used to accomplish this, the result is that it reduces the performance of the entire network. [13]

✚ One more disadvantage arises when the code size of IDS is large. This results in slowing the network because whenever the mobile agent has to go round the network, the entire code has to be moved along with it. [13]

11 Port

A computer that is connected to a network will most likely have open ports. Ports “are used in the TCP [RFC793] to name the ends of logical connections which carry long term conversations.” [1] For one computer to talk to another, one computer must have an open port that is listening for the other computer to connect to. The destination computer will have one or more open ports that the source computer knows about in order to make a connection. There is a maximum of 65535 open TCP (Transmission Control Protocol) ports on a TCP/IP connected computer. There are also, thousands of UDP (User Datagram Protocol) ports that are available. The Internet Assigned Numbers Authority (IANA) has broken down these port numbers into three categories:

- Well-Known Ports are those from 0 through 1023
- Registered Ports are those from 1024 through 49151
- Dynamic and/or Private Ports are those from 49152 through 65535

Well-Known ports are used by system processes, or those requiring special privileges. These are the most common ports that network services run on-and the most popular targets for attack. The IANA is responsible for maintaining the Well-Known ports. The second group, the Registered Ports, is a large group of loosely based ports which have no enforced restrictions, which is also true for the Dynamic/Private ports. The Well-Known ports are also known as Standard ports, whereas any port above 1023 is known as a Non- Standard port. Opening a Standard port does require special system privileges, but any program can open an unallocated Non-Standard port. While it is true that the IANA is an attempt at standardization, almost any service on a computer can be run on any port. [29]

12 TCP/IP, ports and services

12.1 TCP/IP background

TCP/IP is a protocol suite whose crucial precursors were initially developed in early 1960. Cerf, V.G. and Kahn, R.E. published in 1974, the famous paper *A Protocol for Packet Network Interconnection* where they describe the *Transport Control Protocol (TCP)*. [21] The last version of the TCP definition is found in a so-called Request For Comment document (RFC). The RFC are documents created by different expert groups. RFC's may often be used as a standard.

Today, the TCP/IP protocol suite consists of many protocols. Typically well-known protocols that use TCP as the base engine are; HTTP (TCP port 80), SMTP (TCP port 25) and FTP (TCP port 21). There has been an enormous evolution in communication protocols since this early stage. Still, there are several protocols that have been rather unchanged in many decades and that continue to produce online services 24/7/365 throughout Internet. The basic protocols that were originally developed in 1970s, were not built for security, but with functionality as main focus. In order to increase the level of security, we typically add security mechanisms such as IDS.

12.2 TCP flags and their usage

The following Figure 2.2 shows the header of a TCP packet. The two fields; (i) *reserved* and (ii) *flags* are of most interest. The flag field is used to control the TCP connection. Originally, the RFC-793 defined the reserved and the flag field both 6-bits long. Resent RFC's have used 9 bits for the flag field at the expense of the reserved field (RFC-3168; (i) ECN flag (Explicit Congestion Notification Echo) + (ii) CWR flag (Congestion Window Reduced) and RFC-3540; (iii) NS flag (Nonce Sum). These new flags are used to control (or avoid) congestion. [21]

Source port		Destination port	
Sequence number			
Acknowledgment number			
Data offset	Reserved	Flags	Window
Checksum		Urgent pointer	
Options (+ padding)			
Data (variable)			

ECN	CWR	URG	ACK	PSH	RST	SYN	FIN
-----	-----	-----	-----	-----	-----	-----	-----

Figure 2.2 TCP Header - Flags.

We will briefly each flag and its purpose. Much could be written about each flag, but a brief account will be provided. [32]

12.2.1 Explicit Congestion Notification

It is only used by ECN-TCP connections. The ECN-Echo is sent to inform the other side of the TCP connection that it received an ECN notification from the network. This may trigger a TCP Slow Start. Like the CWR field, this is rarely seen in connections. More detail is available in RFC 3168.

12.2.2 Congestion Window Reduced

It is described in RFC 3168. It is only used in TCP connections where Explicit Congestion Notification is used and rarely seen in most TCP Conversations. ECN allows routers to inform the TCP endpoints that their transmit buffers are filled due to congestion. This allows the TCP endpoints to slow their data transmission to prevent packet loss. The CWR field is set by the sender to show that it received a TCP segment with the ECN flag set. [32]

12.2.3 Urgent

It is used to indicate that “urgent” data is carried in the packet. The receiving TCP Stack can process the urgent data in this packet immediately. [32]

12.2.4 Acknowledgement

It is used to indicate that data has been successfully received. It is also used when establishing and tearing down TCP connections. In many cases, every packet in a TCP connection has this flag checked after the initial SYN. [32]

12.2.5 Push

It is often set at the end of a block of data and when signaling the receiver to process the block of data. This bit can be used to monitor the application blocks on the sending application. [32]

12.2.6 Reset

This bit is used to inform the receiver that the sender has shut this connection down. A reset is an abrupt way to do this, but may be legitimately seen at the end of some TCP connections. Watch for these when experiencing application disconnects. [32]

12.2.7 SYN

This bit is used at the start of the TCP handshake to establish the connection. [32]

12.2.8 FIN

It is used to gracefully tear connections down. Each side of the connection sends a FIN, is followed by an ACK, then the connection is finished. [32]

12.3 TCP Connection Establishment

TCP is a connection-oriented, reliable transport protocol. Connection-oriented means the two applications using TCP must establish a TCP connection with each other before they can exchange data. Reliability is provided in TCP by the use of checksums, timers, data sequencing and acknowledgments. By assigning a sequence number to every byte transferred, and requiring an acknowledgment from the other end upon receipt, TCP can guarantee reliable delivery. Sequence numbers are used to ensure proper ordering of the data and to eliminate duplicate data bytes. Note that in a TCP session there are usually two streams of data (every end point is receiving from one of them and sending through the other). So an *ISN* (Initial Sequence Number) must be assigned to each stream when the connection is being established. To see how it is done, let us suppose that C is a client wishing to connect to the server S, and analyze the connection establishment process (often called the *three-way handshake*): [14]

1. C sends a TCP message (known as SYN request) to S with the special flag SYN (SYNchronize sequence numbers) set to ON. A SYN request specifies the port number of the server that the client wants to connect to, and the client's ISN (XX in this example).
2. The server responds with its own SYN message containing S's ISN (YY) and acknowledging C's SYN by specifying that the next byte expected from C is the byte numbered XX+1.
3. C acknowledges the SYN message from S, and data transfer may take place.[14]

13 Port scanning

Port scanning is a technique used by attackers, curious individuals, and administrators to collect information from computers connected to a network. System and network administrators use port scans to identify open ports to a system so that they may be able to limit access to those ports, or shut them off entirely. Attackers use port scanning in the same way that administrators do, but with malicious intent. There are many techniques used in stealth scanning, ranging from those that prevent being detected by logging systems, identity concealment, to confusing the server with invalid information. Each one of these techniques is interesting in its implementation and execution. [29]

14 Types of port scans

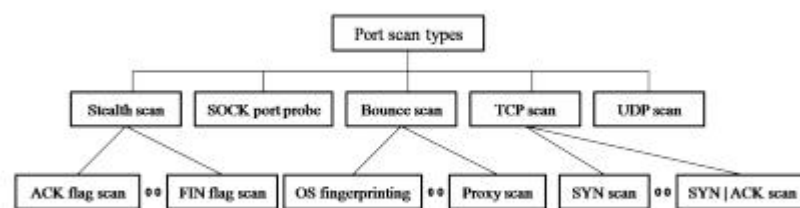


Figure 2.1 Types of port scans

14.1 Stealth Scan

Such a scan is designed to go undetected by auditing tools. It sends TCP packets to the destination host with stealth flags. Some of the flags are SYN, FIN and NULL. [15]

14.2 SOCKS Port Probe

A SOCKS port allows the sharing of Internet connections on multiple machines. Attackers scan these ports because a large percentage of users misconfigure SOCKS ports, potentially permitting arbitrarily chosen sources and destinations to communicate. A SOCKS port on a system may allow the attacker to access other Internet hosts while hiding his or her true location. [21]

14.3 Bounce Scan

It takes advantage of a vulnerability of the FTP protocol itself. Some applications that potentially allow bounce scans are email servers and HTTP Proxies. [15]

14.4 TCP Scanning

TCP connection is never fully established during this type of scanning. So, it is used by smart attackers. If the attacker can clearly know that a remote port is accepting connections, the attacker can launch an attack immediately. It is much more difficult for network defenders to detect since this kind of connection attempts are not logged by the server's logging system. Some TCP scans are TCP *Connect()*, Reverse Identification, IP Header Dump scan, SYN, FIN, ACK, XMAS, NULL and TCP Fragment. [15]

14.5 UDP Scanning

It attempts to find open ports related to the UDP protocol. However, UDP is a connectionless protocol and thus, it is not often used by attackers since it can be easily blocked. [15]

15 Slow scan

Another approach to stealth scanning is the slow scan. This is a low-tech solution to the problem of being logged or noticed by the remote system. A “normal” scan will go through thousands of ports within a short time frame, usually under a minute. The nature of TCP *connect()* scans allows them to be easily logged and discovered by the system administrator. While scanning very quickly can fill a log with information, slow scanning becomes a very viable option. By waiting for a given amount of time between scans for individual ports, logging programs can be defeated. The downside to this type of stealth is the time factor involved. To be stealthy enough to be undetected by an intrusion detection system or a system administrator can take a very long time. For example, a port scanner is set to scan one host

with ports ranging from 1-1024 and given a time lapse of 5 minutes between each port, will take approximately 85 hours. There is nothing fancy about this method, but it does prove that unless a history is kept of all the attempts to each port, detection becomes very difficult. [29]

In other words, slow port scanning is performed for a reason: to avoid being detected by monitoring systems like NIDS. Slow port scanning is the stealthy way of performing a port scan.

Detecting slow port scanning is in general just a question of counting unsuccessful TCP connections over time. In addition to this, we must record odd handshake behavior and non-standardized protocol usage. Slow port scanning may not be difficult to detect - it is more a question of security level versus many hours and equipment expenses. Slow port scanning described e.g. a port scan from a victim and/or an IDS point of view is executed slower than today's normal performance of hardware and software. The slow scanning can typically scan ports and/or hosts in an interval (frequency) of 1, 10, 60, 300 seconds or even slower. [21]

16 Approaches to port scanning

Based on how scanning is performed, port scan techniques can be classified into two broad categories: *single-source* port scans and *distributed* port scans. Each of these categories is illustrated in Figure 2.2 and discussed next.

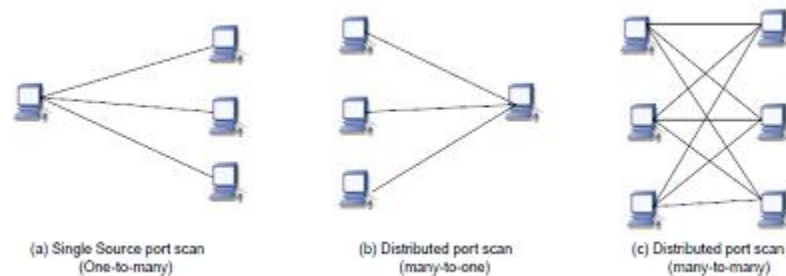


Figure 2.2 Single-source and Distributed port scans

16.1 Single-source Port Scans

In the literature, a port scanner is defined as consisting of “*specialized programs used to determine what TCP ports of a host have processes listening on them for possible connections*”. [15] Staniford et al. [15] further define scan footprint as the set of ports or IP combinations that the attacker is interested in characterizing. According to them, port scans can be of four types (shown in Figure 3): *vertical*, *horizontal*, *strobe* and *block*. A *vertical* scan consists of a port scan of some or all ports on a single computer. The other three types of scans are used over multiple IP addresses. A *horizontal* scan is a scan of a single port across

multiple IP addresses. If the port scan is of multiple ports across multiple IP addresses, it is called a *strobe* scan. A *block* scan is a port scan against all ports on multiple IP addresses. [15] quantified vertical and horizontal scans, defining a vertical scan as consisting of six or more ports on a single computer, and a horizontal scan as consisting of five or more IP addresses within a subnet.

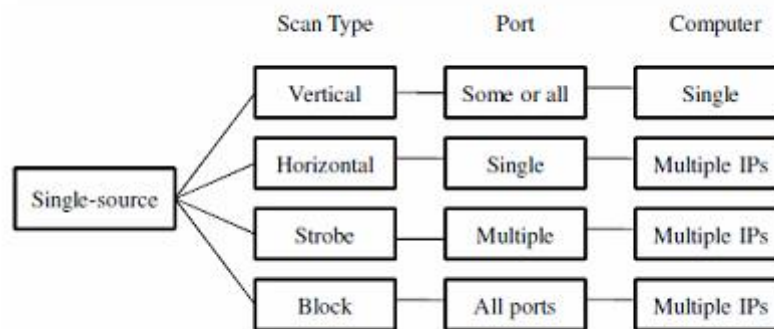


Figure 2.3 Single-source scan types with its ports details

16.2 Distributed Port Scans

Distributed information gathering is performed using either a *many-to-one* or a *many-to-many* model. [15] Here, the attacker utilizes multiple hosts to execute information gathering techniques in different ways: *rate-limited* and *random* or *non-linear*. In a *rate-limited* information gathering technique, the number of packets sent by a host to scan is limited [15]; this is based on the FreeBSD (BSD-Berkeley Software Distribution) implementation of Unix where separate rate limits are maintained for open ports as well as closed ports. For example: TCP RST rate limited, ICMP port unreachable rate limited, and so on. On the other hand, a *random* or *non-linear* gathering technique refers to randomization of the destination IP-port pairs amongst the sources, as well as randomization of the time delay for each probe packet.

A *coordinated* attack has a more generic form of a distributed scan described by Staniford-Chen et al. [15]. It is defined as multi-step exploitation using parallel sessions with the objective of obscuring the unified nature of the attack or allowing the attackers to proceed more quickly. However, Green et al. [15] define a *coordinated* attack as “multiple IP addresses working together towards a common goal.” They also add that a coordinated attack can be viewed as multiple attackers working together to execute a distributed scan on many internal addresses or services. Staniford et al. [15] later define a distributed scan as one that is

launched from a number of different real IP addresses, so that the scanner can investigate different parts of the footprint from different places. An attacker can scan the Internet using a few dozen to a few thousand zombies. A zombie is a compromised host, whose owner is unaware of the fact that the computer is being exploited (a remote attacker has accessed and set up to forward transmissions (spams or viruses) to other computers on the network) by the external party. Yegneswaran et al. [15] define *coordinated* scans as being scans from multiple sources aimed at a particular port of destinations within a one hour window.

17 Approaches to port scan detection

We classify various port scan detection approaches available in the literature into two different categories: *single-source* approaches and *distributed* approaches. Single-source port scan is performed following either a *one-to-one* or a *one-to-many* model for gathering information about a target computer or network. On the other hand, distributed information gathering [15] is performed using a *many-to-one* or *many-to-many* model for gathering information about a target computer or network. A hierarchy of the scan detection approaches is reported in Figure 4.

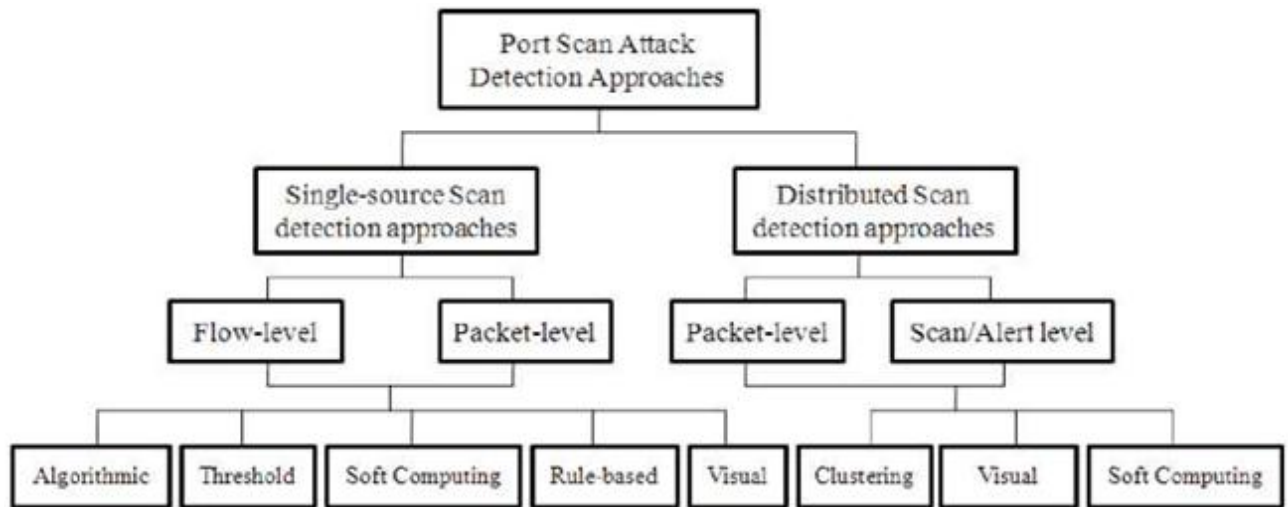


Figure 2.4 Hierarchy of port scan attack detection approaches

17.1 Single-Source Port Scan Detection Approaches

Detection approaches for single-source port scans have been part of intrusion detection systems since 1990, from the release of Network Security Monitor (NSM) [15]. We divide these detection approaches into five categories: Algorithmic, Threshold-based, Soft Computing based, Rule-based, and Visual. Each of these can be further categorized based on the type of network data processed, methodology used for detection and the evaluation

criteria. For example, some approaches exploit packet level information whereas some others use flow level information. These details provide not only the connection information, but also allow one to analyze the packet payload. This allows signatures of known attacks to be used on the data to determine whether or not the packet payload contains an attack. Flow level information is provided by Cisco NetFlow [15] and Argus [15] in the form of summarized connection information. [15]

17.1.1 Algorithmic Approaches

These approaches use methods such as hypothesis testing and probabilistic models, to detect port scan attacks based on analysis of network activity. [15]

17.1.2 Threshold-based Approaches

These approaches examine events of interest X across a Y -sized time window to detect port scan attacks above certain thresholds [15]. The most commonly used parameter for detecting scans is the number of unique IP addresses contacted by a host. Several intrusion detection systems have been developed in the past couple of years in the public domain that use the threshold-based approach to detect anomaly. The approach requires the packet level information. [15]

17.1.3 Soft Computing Approaches

Soft computing includes important methods that provide flexible information processing for handling real-life ambiguous situations [15]. Methods in soft computing exploit tolerance for imprecision and uncertainty, use approximate reasoning and partial truth in order to achieve traceability, provide robustness and low-cost solutions to problems. [15]

17.1.4 Rule-based Approaches

Generally, a rule-based IDS analyzes traffic data passing through it and differentiates intrusive traffic behaviors from the normal. A rule-based IDS uses rules stored in its knowledge base to detect and take actions when anomaly occurs in the traffic or when there are unauthorized activities. A rule-based IDS must generate rules based on network activity for detecting anomaly. [15]

17.1.5 Visual Approaches

Some approaches present data to the user in a visual manner so that he or she can

recognize scans by the patterns it generates. Such approaches detect and investigate port scans using packet level information, and flow level information. [15]

17.1.6 Discussion

A large number of techniques for detection of port scans have been reported in this section under five distinct categories of approaches. However, it is not always easy to unambiguously classify a technique into any one of these approaches since it often uses elements from multiple classes. These approaches use features such as the source IP and port, destination IP and port, protocol, start time and end time of the session, and the number of bytes, and packets transferred. [15]

17.1.7 Distributed Scan Approaches

The main goal of these approaches is to detect coordinated attacks. These types of attacks attempt to compromise a single host from multiple systems. There are various methods for detecting these attacks. Like the single-source scan detection approaches, based on the features used by the methods, the approaches also can be categorized into four classes: Algorithmic, Clustering, Soft Computing, and Visual.

17.1.8 Algorithmic Approaches

Only few algorithmic approaches that operate in a distributed mode can be found in the literature. We report here two popular techniques GATES and WHITE which have been established to perform satisfactorily over multiple datasets. [15]

17.1.9 Clustering Approaches

Clustering is the process for partitioning data into groups of similar objects. It is an unsupervised learning process. There are many approaches available for detecting network scans based on the similarity of the data, compactness of the cluster. [15]

17.1.10 Soft Computing Based Approaches

In addition to those approaches discussed so far, there are several distributed scan detection approaches, that use soft computing techniques such as **Curtis et al, Zhang et al.** [15]

17.1.11 Visual Approaches

These approaches are used for visualizing network traffic to detect whether the flow of network packet is an attack or normal behavior. One such commonly found approach is proposed by Conti and Abdullah [15]. The approach attempts to detect distributed scans

against a background of normal traffic based on visualization. Due to the lack of details, it is difficult to understand how a distributed scan would use this tool. Also, it is not clear how much traffic can be viewed at one time without obscuring features of interest. [15]

17.1.12 Discussion

Most distributed port scan detection approaches analyze packet level information. They can detect port scan attacks based on the IP addresses (source IP, destination IP), connection information, port (source ports, destination ports) fields in the IP header. [15]

17.1.13 Port Scanning techniques

The simplest way of performing port scanning is to try to connect to every TCP port from 1 to 1023 on the victim.

A normal TCP connection is done by using the sequence shown leftmost in figure 2.5 We complete all three sequences: SYN + SYN/ACK + ACK.

A SYN Scan is shown rightmost in figure 2.5 This is not a complete sequence to establish a TCP session, and is therefore called a SYN scan. This scan may not be logged or registered by security systems if this is performed in a stealthy way (slowly). [21]



Figure 2.5 Normal TCP sequence (left) and SYN Scan (right).

There exist a lot of scans. We categorize port scans into the following list:

- Open Scan
- Half Open Scan
- Stealth Scan
- FTP Bounce Scan
- Fragment Packet Scan
- UDP Scan

Port scanning may also be used in an exhausting/denial of service (DoS) attack. This is terrifying easy to perform, but generates (of course) a lot of noise (alarms/logs etc.).

Port scanning may also give the scanner much more information than the status of the actual port (open, closed or filtered (behind a firewall port)). Port scanning may also unveil the vendor, the operating system, the version of the application (that answers on the port) etc. [21]

18 Nmap

There are plenty of free scanning tools like Angry IP, Superscan, NetScanTools, Unicornscan ..ect however Nmap is a scan tool that offers all the features mentioned in the previous tools.

Nmap ("Network Mapper") is a free and open source ([license](#)) utility for network discovery and security auditing. Many systems and network administrators also find it useful for tasks such as network inventory, managing service upgrade schedules, and monitoring host or service uptime. Nmap uses raw IP packets in novel ways to determine what hosts are available on the network, what services (application name and version) those hosts are offering, what operating systems (and OS versions) they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics. It was designed to rapidly scan large networks, but works fine against single hosts. Nmap runs on all major computer operating systems, and official binary packages are available for Linux, Windows, and Mac OS X. [34]

18.1 Commonly used Scans

The following attacks are very often seen. We will now describe each of these scans: [30]

Feature	Option
TCP SYN Scan	-sS
TCP Connect Scan	-sT
UDP Scan	-sU
TCP NULL Scan	-sN
TCP FIN Scan	-sF
Xmas Scan	-sX
TCP ACK Scan	-sA
Custom TCP Scan	-scanflags
IP Protocol Scan	-sO

Table 2.1 Commonly used TCP Scans.

18.1.1 TCP SYN Scan (-sS)

The TCP SYN scan is the default option for privileged users (users running as root on Unix/Linux or Administrator on Windows). The default TCP SYN scan attempts to identify the 1000 most commonly used TCP ports by sending a SYN packet to the target and listening for a response. This type of scan is said to be stealthy because it does not attempt to open a full-fledged connection to the remote host. This prevents many systems from logging a connection attempt from your scan. [30]

18.1.2 TCP Connect Scan (-sT)

The -sT scan is the default scan type for non-privileged users. It is also used when scanning IPv6 targets. The TCP Connect Scan is a simple probe that attempts to directly connect to the remote system without using any stealth. [30]

18.1.3 UDP scan (-sU)

Many network services (like DNS, DHCP, and SNMP) still utilize UDP. When performing a network audit, it is always a good idea to check for both TCP *and* UDP services to get a more complete picture of the target host/network. [30]

18.1.4 NULL Scan (-sN)

A TCP NULL scan causes Nmap to send packets with no TCP flags enabled. This is done by setting the packet header to 0. Sending NULL packets to a target is a method of tricking a firewalled system to generate a response. [30]

18.1.5 FIN Scan (-sF)

In a -sF scan, Nmap marks the TCP FIN bit active when sending packets in an attempt to solicit a TCP ACK from the specified target system. This is another method of sending unexpected packets to a target in an attempt to produce results from a system protected by a firewall. [30]

18.1.6 XMAS Scan (-sX)

In the Xmas scan, Nmap sends packets with URG, FIN, and PSH, and flags activated. This has the effect of "lighting the packet up like a Christmas tree" and can occasionally solicit a response from a firewalled system. [30]

18.1.7 TCP ACK Scan (-sA)

The -sA option can be used to determine if the target system is protected by a firewall. When performing a TCP ACK scan, Nmap will probe a target and look for RST responses. If no response is received the system is considered to be filtered. If the system does return an RST packet, then, it is labelled as unfiltered. [30]

18.1.8 Custom TCP Scan (--scanflags)

The --scanflags option allows users to define a custom scan using one or more TCP header flags. Any combination of flags listed in the table below can be used with the --scanflags option. For example: nmap --scanflags FIN ACK (no space) would activate the FIN and ACK TCP flags. [30]

18.1.9 IP protocol scan (-sO)

The IP protocol scan displays the IP protocols that are supported on the target system. The most commonly found protocols on modern networks are ICMP, TCP, and UDP. Using the -sO option is helpful for quickly identifying what types of scans you want to perform on the selected target system based on its supported protocols. [30]

We summarize the different scans and the possible outcomes in Table 2.2 [21]

PST*	Protocol	TCP flag	VR*(OP*)	VR*(CP*)	FLDP*
TCP <i>Connect()</i>	TCP	SYN	ACK	RST	Yes
Reverse Ident	TCP	No	No	No	No
SYN Scan	TCP	SYN	ACK	RST	Yes
IP Header Dump Scan	TCP	No	No	No	No
SYN ACK Scan	TCP	SYN ACK	RST	RST	Yes
FIN Scan	TCP	FIN	No	RST	No
ACK Scan	TCP	ACK	No	RST	No
NULL Scan	TCP	No	No	RST	No
XMAS Scan	TCP	All flags	No	RST	No
TCP Fragment	TCP	No	No	No	No
UDP Scan	UDP	No	No	Port Unreachable	No
FTP Bounce Scan	FTP	Arbitrary Flag Set	No	No	No

Table 2.2 Details of port scan types and their firewall level detection possibilities.

Note: PST (Port Scanning Technique), VR (Victim's Reply), OP (Open Port), CP (Closed Port), FLDP (Firewall Level Detection Possibility).

19 Conclusion

Intrusion detection is currently attracting interest from both the research community and commercial companies. In this paper, we have given an overview of Port scanning, we attempted to summarize as much as we could because it takes a book to mention all the scans like UDP or TCP scans and the different algorithms to detect these scans. In this chapter, we dealt with port scanning, its technique, types and-tools. We chose the distributed port scan which is many to one model as a Scan type and Nmap as a scan tool in our implementation like we will see in next chapter.