Indian Institute of Technology Kanpur

CS425: Computer Networks

Port Scanning Utility

Group 15

Sourav Khandelwal (sourav@iitk.ac.in)

Anurag Awasthi (anuraga@iitk.ac.in)

Vismay Chintan (vismay@iitk.ac.in)

Abstract

Essentially, port scanning involves sending a message to each port, one at a time. The kind of response received indicates whether the port is in use and can therefore be probed for weakness. Port scanning has legitimate uses in managing networks as used by the crackers as well as can be malicious in nature if some hackers are looking for a security breach in the computer system on the network. Some examples of port scanners or port scanning tools are NMap, Foundstone Vision and Portscan 2000. Among them, NMap (current version: 4.76) claims the actual standard in the security industry due to its all-round capabilities in port scanning. Here in our project, we have implemented a port scanning utility which can perform the different scans and gives standard results about the port states, services running and IP hosts. The implementation of the utility is done in "Python" programming language and various functions of the python tool scapy is used for the construction of packets and other networking functionalities. We describe various port scanning utilities and standard results demonstrating the success of our utility for scanning the ports.

Acknowledgements

We would like to thank our instructor Prof. RK Ghosh for encouraging us to work in this fast developing field of Networks Security. Thanks to Dr. Ghosh, we have been able to appreciate the applicability of Port Scans and network Vulnerabilities in Network Systems.

Introduction

Port Scanning is one of the most popular techniques attackers use to discover services that they can exploit to break into systems . All systems that are connected to a LAN or the Internet via a modem run services that listen to well-known and not so well-known ports. By port scanning the attacker can find the following information about the target systems:

- *The host or hosts are alive on the target network*
- *What Services are running?*
- *What users own those services?*
- *The port state of target machine ports.*

Port scanning is accomplished by sending a message to each port, one at a time. The kind of response received indicate whether the port is used and can be probed for further weakness. Port scanners are important to network security technicians because they can reveal possible security vulnerabilities on the targeted system.

***Ports and Services*:**

There are hundreds of ports and services registered with the Internet Assigned Number Authority(IANA). In practice, less than one hundred are in common use.

The port numbers are divided into three ranges:

Well Known Ports are those from 0 through 1023.

The *Registered Ports* are those from 1024 through 49151.

The *Dynamic* or *Private Ports* are the ports onwards.

Services have assigned ports so that a client can find the service easily on a remote host. For example, telnet servers listen at port 23, ssh on port 22, HTTP on port 80, and SMTP (Simple Mail Transport Protocol) servers listen at port 25. Client applications, like a telnet program or mail reader, use randomly assigned ports typically greater than 1023.

Port States:

Six Different Port States are recognized:

- *open*:

  An application is actively accepting TCP connections, UDP datagrams or

SCTP associations on this port. Finding these is often the primary goal of port scanning. Security-minded people know that each open port is an avenue for attack. Open ports are also interesting for non-security scans because they show services available for use on the network.

- *Closed*:

  A closed port is accessible, but there is no application listening on it. They can be helpful in showing that a host is up on an IP address and as part of OS detection

- *filtered*:

  Filtering prevents its probes from reaching the port. The filtering could be from a dedicated firewall device, router rules, or host-based firewall software. Filters generally drop the packets without responding. These ports cause the scan to try again and again, thus slows down the scanning.

- *open | filtered*

  This state is arrived when the scan is unable to determine whether the port is open or filtered. This occurs for scan types in which open ports give no response.

- *closed | filtered*
  - This state is arrived when the scan is unable to determine whether a port is closed or filtered.

Objective

The aim of our project is to build a port scanning utility which can scan the ports of the target systems giving the information about the target hosts, the listening ports, the filtered ports, and the services running on the ports. We aim to achieve the scanning of the ports by implementing the different port scanning techniques discussed in the standard tool for port scanning, Nmap.

Specifications

The language used in the code is Python. A python tool scapy is used for package manipulation functions. A short description of the features implemented and programming interface is provided.

Features:

- **Connect_Scan** technique is implemented to scan the ports by establishing a *TCP connection* through a Socket
- **Ping Scan** is implemented to find the alive ports in the network traffic.
- **SYN_Scan** technique is implemented to determine the state of the ports using a *TCP packet*.
- **UDP_Scan** technique is implemented to determine the state of the ports using a *UDP packet*.
- *FIN_Scan* technique is implemented to determine the state of the ports using a TCP packet with the flags set as FIN.
- Multiple packets are send to scan multiple networking IPs and ports over the network.
- For reliability, few packets are send with the false source to prevent detection of the filters running on the target.
- For avoiding packet loss due to **congestion**, an option for inter packet delay is given between every packet.
- A timeout is defined for the packet to arrive from the target.

Programming Interface:

- ***Scapy*** interface in python has been used to create TCP/UDP/ICMP packets and to send/receive the packets over the network.
- Option parser has been used for I/O in python.

Design

The five major scanning techniques are Ping for alive hosts, Connect Scan, SYN Scan, UDP Scan, FIN Scan. The approach and Logic to develop the scans is described below.

Port Scanning Approaches

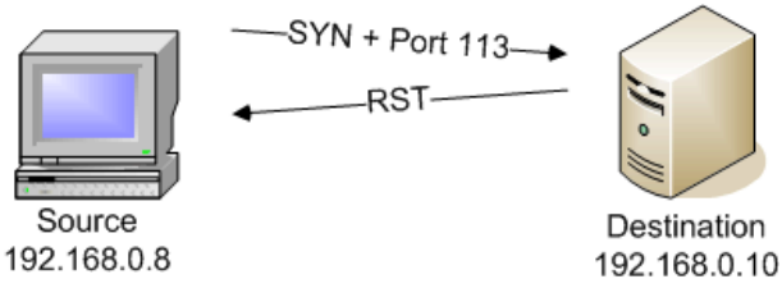
TCP SYN Scan

The TCP SYN scan uses common methods of port-identification that allows to gather information about open ports without completing the TCP handshake process. When an open port is identified, the TCP handshake is reset before it can

be completed. This technique is often referred to as "**half open**" scanning.

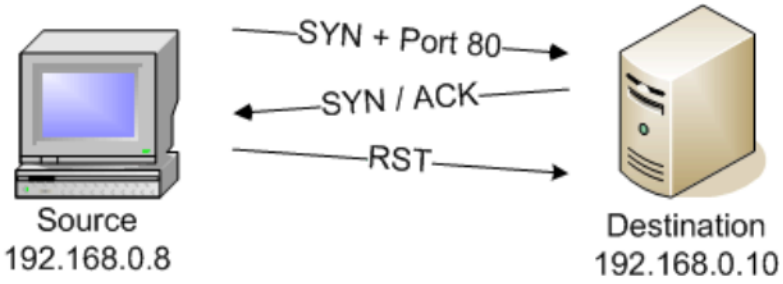
Approach

- Packets with the destination IP, Ports to be scanned specifying the SYN flags in the TCP header is send and received by the host.
- The received packets are analyzed with their flags associated in the TCP header.
- **Closed Ports**: Sends a RST ACK packet in reply to a SYN packet.



Packet	SRC IP > DST IP	Flags	Packet	SRC IP > DST IP	Flags
IP/TCP	172.27.22.28:ftp > 172.27.22.35:auth	S ==>	IP / TCP	172.27.22.35:auth > 172.27.22.28:ftp	RA

- **Open Ports:** Sends a SYN ACK packet in reply to a SYN packet.



Packet	SRC IP > DST IP	Flags	Packet	SRC IP > DST IP	Flags
IP/TCP	172.27.22.28:ftp > 172.27.22.35:www	S ==>	IP / TCP	172.27.22.35:www > 172.27.22.28:ftp	SA

- **Filtered Ports:** Sends a **bad Check Sum** along with the Packet and if the packet is received correctly, then the port is filtered and the filter is automatically replying to all the packets.

Advantages:

- The TCP SYN scan never actually creates a TCP session, so isn't logged by the destination host's applications.

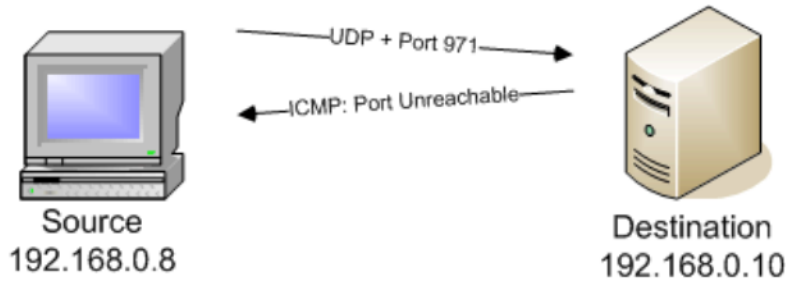
- Quieter Scan and less visibility in the destination system.

UDP Scan

UDP has no need for SYNs, FINs, or any other fancy handshaking. With the UDP protocol, packets are sent and received without warning and prior notice is not usually expected.

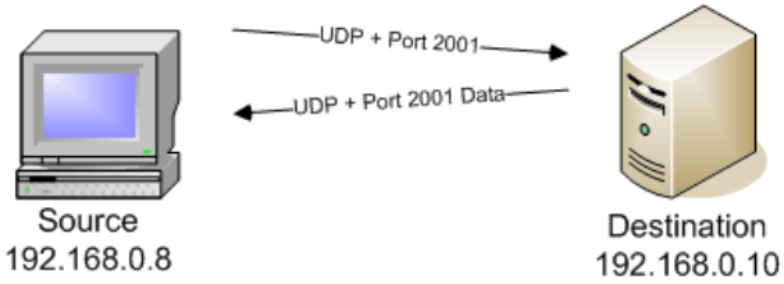
Approach

- A UDP packet is send to the target host and received packets are parsed.
- **Closed Ports:** Target Host responds with an ICMP port unreachable.

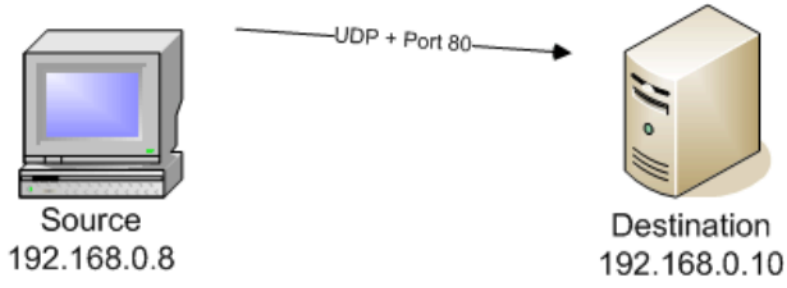


IP/UDP 172.27.22.28:domain > 172.27.22.35:971 ==> IP/ UDP 172.27.22.35>172.27.22.28 dest-unreach 3

- **Open Ports:** if the target responds with the UDP packet from the same dport, then it is considered to be open.



- **Open | Filtered:** if the target does not respond to the packet, it is considered to be filtered.

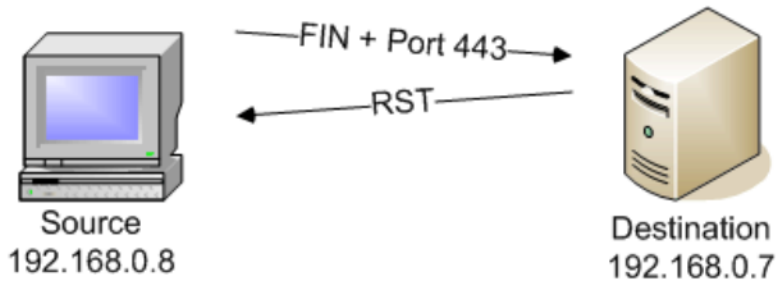


FIN Scan

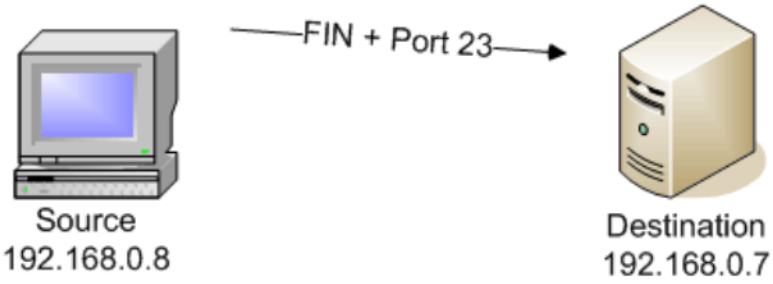
The scanner sends a FIN packet, which should close a connection that is open. Closed ports reply to a FIN packet with a RST. Open ports, on the other hand, ignore the packet in question. This is required TCP behavior.

Approach

- Sender sends a TCP packet to the target host with the FIN flag set in the TCP header.
- **Closed Ports**: The target host sends a RST frame response to the FIN packet.



- **Open Ports:** If a port is open, the target does not respond to FIN packet.



Advantages:

- A stealth scan which are able to decide the state of the ports for which firewalls and packet filters watch for SYNs to the restricted ports.

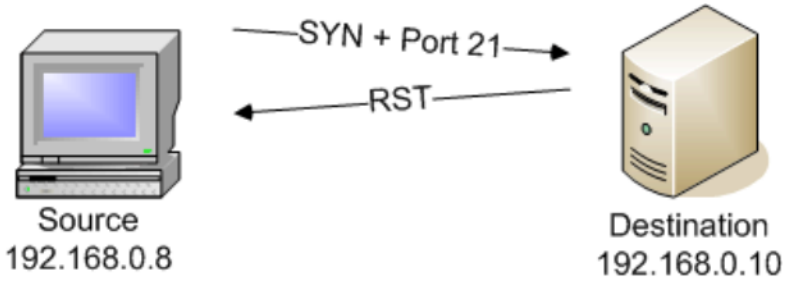
Connect Scan

This is the most basic form of TCP scanning. The connect() system call provided by your operating system is used to open a connection to every interesting port on the machine. If the port is listening, connect() will succeed, otherwise the port isn't

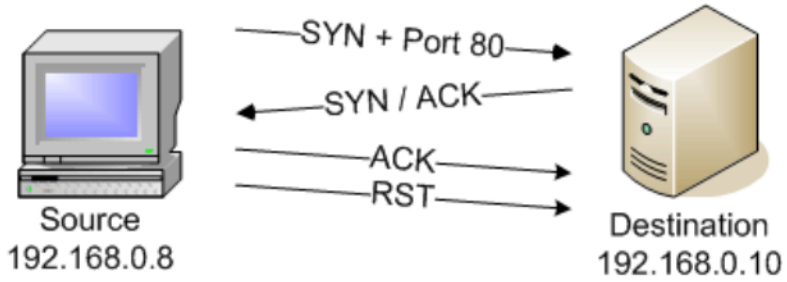
reachable.

Approach

- The Scan method uses the **TCP 3-Way handshake** protocol connection that every other TCP based application uses on the network.
- Make a TCP Socket for establishing Connection.
- Set a particular timeout and try establishing connection to the target with the connect call on socket.
- **Closed Ports:** Error occurs in establishing connection.



- **OPEN Ports:** if Establishes connection through a 3-way protocol.



- After the connection is made by the three way protocol, it is immediately closed.

Advantages:

- Do not need special privileges. Any user can employ this scan.

Implementation & Testing

Implementation

1. Put the IP_RANGE input into the input file for specifying the IP range to be scanned.

2. Start the Program to scan the ports.

Admin privileges are required for all the scans except Connect Scan.

sudo python PortScanner.py -e input_file -<Scantype> -p ports -o outputFile -n noscanfile

Option	Type	Name	Description
-o	Output file	“outfile_”+scanfile	output file where scan results are written.
-e	Expand file	expanded_'+<expand_file>	input file with host ips to be expanded. writes a file 'expanded_'+<expand_file> that is used for scanning
-p	Range	ports	Lists the ports given by the option parameter
-t	interger(sec)	Timeout	timeout to apply to the packets
-g	Interval gap	interval	Interval between sending the packets
-n	noscanfile	noscan	hosts already scanned or not to be scanned with host expanded and ready to be scanned
-c	port-scan	connect_scan	scan the list of ports and ips with TCP Connect Scan
-u	port-scan	udp_scan	scan the list of ports and ips with UDP Scan
-v	port-scan	verify_hosts	scan the list of hosts to verify which are up and which are not by TCP Ping method
-s	port-scan	syn_scan	scan the list of ports and ips with SYN Scan
-f	port-scan	fin_scan	scan the list of ports and ips with FIN scan method

Output Format

IP HOST_UP PORT SERVICES PORT_OPEN FIREWALL

IP - The scanned IP address

HOST_UP – The scanned IP target is alive or not.

PORT- Port scanned

SERVICES- The default service running on the port

PORT_OPEN- Tells whether a port is open or close.

FIREWALL- Tells whether a port is filtered or not.

Limitations

- Connect scan has not been implemented using raw packets at the kernel level.
- FIN scan can not distinguish between open and filtered ports.
- FIN scan does not work on Windows platform.

Future Work

- IP spoofing or Idle Scan
- Scan Detection

References

1. **Nmap Reference Guide**:http://nmap.org/book/man.html
2. **The Online Resource for Network Professionals**: http://www.networkuptime.com/nmap/index.shtml
3. **Scapy**: Powerful interactive packet manipulation :http://www.secdev.org/projects/scapy/