Global Information Assurance Certification Paper

How Does Network Security Scanning Work Anyway?

Ronald Black

September 27, 2000

Introduction

At this point, most Information System Security professionals, network and system administrators have at least heard of port scanners and other network based vulnerability assessment tools. There are many commercial scanners (ISS Internet Scanner, Axent NetRecon, Cisco NetSonar, NAI CyberCop Scanner; just too name a few) and many freeware/open source type products (Nmap, Netcat, Hping, Strobe, SATAN, SARA, Whisker , Saint and many others). In fact, a recent survey found at www.insecure.org/tools.html lists over 10 scanners in its Top 50 Security Tools (which doesn't include Nmap). Of course this survey is based on folks that have and use Nmap. These products run on a variety of operating systems including most flavors of Unix, Linux and even Micosoft Windows NT/2000. So if you haven't already, "**you gotta get you some**" (as the country western song says). If you've never scanned yourself, you may **literally not know what you're missing…..**

Every professional that takes themselves and their job seriously should have an assortment of security tools in their kit. Network scanners are essential for the well dressed professional. They are also an important component in a "defense in depth" architecture. Features depend on particular products, but some of the most common include active host identification, network mapping, port scanning, and operating system identification. Many include features which are geared to defeat other security tools such as audit logs, firewalls and intrusion detection systems. Some identify specific service or platform vulnerabilities and provide information geared to help you "fix" the vulnerability. All provide you with a view of your network and system defenses that are hard to find anywhere else.

In order to understand how network security scanners work, you must understand the basic operation of the TCP/IP protocol suite. Since a full explanation of IP is beyond the scope of this paper, I'll assume that the reader is familiar with what IP, TCP, UDP, ICMP and service ports are (you know, Internet Protocol, Transport Control Protocol, User Datagram Protocol, and Internet Control Message Protocol). Just as a quick review TCP is the reliable, connection oriented protocol that works based on a three-way handshake (SYN, SYN/ACK, and ACK). UDP is the unreliable, connectionless one and ICMP is for error reporting, gathering network information, flow control, and packet rerouting. The Ping and Traceroute (Tracert) utilities use ICMP. These all ride on top of IP. Service ports are used by the destination computer to connect to the appropriate service daemon (FTP, HTTP, Telnet, etc). The combination of source IP address - service port and destination IP address - service port identify the connection between two systems in a TCP connection. When dealing with TCP (along with the SYN & ACK) you have reset (RST) and finish (FIN) which play an important role in scanning. OK? Well, we're off on our journey and if you're already lost you need to adjust the frequency on the transmitter. Or just say "Beam me up, Scottie".

Basic Scanning Techniques

Fyodor[1], the author of Nmap, identifies 9 basic techniques in his work, "The Art of Port Scanning". They are (and I quote):

- TCP connect() scanning,
- TCP SYN (half open) scanning,
- TCP FIN (stealth) scanning,
- TCP ftp proxy (bounce attack) scanning
- SYN/FIN scanning using IP fragments (bypasses packet filters)
- UDP recvfrom() scanning,
- UDP raw ICMP port unreachable scanning,
- ICMP scanning (ping-sweep), and
- reverse-ident scanning.

OK, that's great but what does all that mean. We'll get into the details in a minute. Fyodor also discusses means of identifying remote operating systems through differences in how each Operating System vendor implements its TCP/IP stack in another work[2]. Ofir Arkin also discusses these basic scanning techniques in two works (http://www.sys-security.com/html/papers.html) and also discusses techniques in identifying remote OS. The same techniques used to identify remote OS's in active scanning can also be used passively. But the devil is in the details and we're ready to begin.

ICMP Scans

As previously mentioned, Ping and traceroute are ICMP based utilities or applications. Most administrators find these very useful tools. And the same basic capability is usually built into network security scanners. Using ICMP Echo requests (ICMP type 8); the scanner can perform what is known as a Ping Sweep. Scanned hosts will reply with an ICMP Echo reply (ICMP type 0) indicating that they are alive. No response may mean the target is down or does not exist. ICMP Echo requests can also be sent to a network broadcast address. Using ICMP Echo requests to a network broadcast address can also help in identifying the remote operating system. Windows systems don't respond to ICMP broadcasts while many older Unix implementations still do, particularly if they haven't got the latest patches. Simply blocking ICMP Echo requests at a firewall isn't enough. ICMP type 13 messages (TIMESTAMP) and type 17 (Address Mask Requests) can be used in the same way.[3] Arkin spends considerable effort detailing the dangers of ICMP in his work "ICMP Usage in Scanning or Understanding some of the ICMP Protocol's Hazards". By the way, since ICMP doesn't utilize ports, this is not a port scan technique.

UDP Scans (or UDP raw ICMP port unreachable scan)

This method uses information derived from the receipt (or lack thereof) of an ICMP port unreachable message. The scanner sends a UDP data gram to a UDP port on a target system. If no message is received, then the port might be listening. This type of scan has multiple variables and isn't that reliable but it is still used. Fyodor points out that the Solaris rcpbind service (possible exploit) can be found with UDP scans.

TCP Scans ("Where is the Beef?")

The most basic form of scanning is TCP Connect scanning. Do you remember the basic 3 way handshake of TCP? Well, pick a port on an active host and attempt to connect to it. If the port is listening, the connect should succeed. If you don't connect, then the port isn't reachable. Easy isn't it? Now if you connect to well know ports in linear fashion, you are doing a port scan. But this is easily detectable (the system log should record each connect and immediate shutdown). Now we start to get a little more careful. What if we don't open a full connection? What if we send the SYN first and then a RST(reset) if we receive a SYN/ACK? This connection is only half open. We've gotten what we wanted, didn't we. The port is listening because the remote system replied with a SYN/ACK. This is know as TCP SYN scanning or a "half-open" scan. Many systems won't log this so no one will be the wiser (maybe).

As we continue to seek ways to avoid detection, our scanner will most likely throw in something called TCP FIN scanning. What happens when we send a TCP FIN? There has been no connection. Well, the open port (listening port) will probably ignore the packet because it knows there was no connection. The non listening port will probably send a RST (like it does when we're trying to open a connection). So that gives us useful information: no response means the port is listening. A RST may mean that the port is not listening or it may mean that we're dealing with a Windows system. Rather than sending a TCP FIN or a TCP SYN, what happens when we send a TCP SYN/ACK? Arkin says if the system plays by the rules of TCP, a closed port will send a RST as it did with the SYN FIN. If we set all the flags (XMAS or Christmas Tree scan) or no flags (NULL), we are still looking for closed ports to send a RST. Open ports will drop the packets. This activity isn't likely to be logged by the host being scanned so we have a stealth scan. Network Intrusion Detection Systems (IDS) are still likely to pick this up.

TCP FTP proxy (bounce attack) scanning takes advantage of a weakness in RFC 959. This RFC includes support for proxy ftp connections. So I can connect to the protocol interpreter (PI) of an FTP server and request that the server initiate a Data Transfer Process (DTP) to some other system(s). Hobbit, creator of Netcat, pointed out this "feature" as a means of attack in 1995. TCP FTP proxy scanning uses this weakness to scan ports on remote systems. Arkin states "Nmap supports this kind of scan and uses the PORT FTP command to declare that our passive user data transfer process is listening on the target box at a certain port number. We then use the LIST FTP command to try to list the current directory. The result is sent over the server data transfer process channel. If the transfer is successful (150 and 226 response), the target host is listening on the specified port. Otherwise a "425 Can't build data connection: Connection refused" message is sent."[4] Now who do you think is gonna be seen as the scanner? Any and all ports can be scanned by issuing one PORT command after another. Not the quickest scan method but certainly "stealthy". By the way, some FTP servers disable the "proxy" feature so this won't always work on the 1st FTP server you pick. However, persistence will pay off.

Fragmenting IP packets is a variation of other TCP scanning techniques. Instead of sending a single probe packet, you break it into two or more packets which are reassembled at the destination. We will not go into the other techniques here because I want to talk about techniques of identifying remote operating systems. However, I will mention that some scanners include options for spoofed ip addresses. Nmap provides this capability. How? Well it provides an option to define the "source" address of the packet.

Techniques for Identifying the Remote Operating System (OS)

Arkin says "Because many security holes are operating system dependant, identifying which operating system runs on the target host / machine is of major importance."[5] Not a news flash but something that should be incorporated in every scanner. OK, so how does it work. Some services like telnet or ftp make it easy by providing a banner which readily identifies the operating system. Others give themselves away in error message information. And we administrators can provide too much information in a DNS entry. But there are other, less obvious means of identifying a remote operating system. Fyodor calls it TCP/IP stack fingerprinting. This may be a news flash. **Not all vendors implement TCP/IP the same way!** Those differences can help to identify the OS of the system you are scanning. In fact, these differences can be used passively to identify the remote OS on traffic just passing by.[6] Differences in how things like Time To Live (TTL), initial Window Size, the Don't Fragment (DF) bit, and Type of Service (TOS) are implemented can give you a reasonable idea of what the remote operating system might be. For example, did you know that a Windows system sets its initial TTL at 128 while Red Hat Linux sets its initial TTL at 64?[7] By rounding up to the closest variation of 16 or 32, you can estimate the original TTL was. By the way, system administrators can (usually) eliminate banners and change default TTLs. But there is more!

Remember the TCP FIN probe? Well, the correct RFC 793 response is for an open port is not to respond. However, "several implementations such as MS Windows, BSDI, Cisco, HP/UX, MVS, and IRIX send a RESET back. Most current tools utilize this technique".[8] By finding patterns in the initial sequence number during a connection request, you can determine a lot about the operating system. There are many more variations of TCP/IP implementation that will help your scanner identify what OS is running on the target system. You'll need to check out the references.

What's Next?

We've figured out what hosts are alive, what services are running, and what the remote operating system probably is. **So what you say! "I already knew what services ran on my systems and what the operating systems were", you say.** What does that tell us? **It tells us what weaknesses to test for!** Its not going to do us much good to test for a Solaris rcpbind exploit on a Windows NT box. However, some scanners aren't that picky. Some will run all known exploit signatures against any active box. Some have password cracking capabilities. Others, like Nmap, don't have exploit signature or password cracking capabilities. Nmap will, however, estimate how hard it will be to highjack an open session. Some scanners like Whisker are geared to a specific type of

application (in this case, CGI). Just a few brief comments here on CGI scanning and we'll let you investigate further for yourself. We've already established in the scan that http is open on the target box. And we know the operating system. So now we want to check for /cgi-bin. If we query for the cgi-bin directory, we'll get a 200 (ok), 403 (forbidden), or 302 (for custom error pages) if it exists. We'll get a 404 if it doesn't. As you can see, we've established quite a bit already. If it exists, we know if we can get at it (200). Let the hacking begin…..

How do you defend yourself? You start by knowing where you might be vulnerable. Now you can see what anyone on the "net" can see and now you know what they will know about you! And that is an important step in defending yourself.

References

1. Fyodor, "The Art of Port Scanning" September 1997
URL: http://www.insecure.org/nmap/p51-11.txt (Sept 18th 2000)

2. Fyodor, "Remote OS detection via TCP/IP Stack FingerPrinting" April 10, 1999
URL: http://www.insecure.org/nmap/nmap-fingerprinting-article.html (Sept 18th 2000)

3. Arkin, Ofir, "Network Scanning Techniques" November 1999
URL: http://www.sys-security.com/html/papers.html (Sept 18th 2000)

4. Arkin, Ofir, "ICMP Usage in Scanning or Understanding some of the ICMP Protocol's Hazards" September 2000
URL: http://www.sys-security.com/html/papers.html (Sept 18th 2000)

5. Miller, Toby, "Intrusion Detection Level Analysis of Nmap and Queso" August 2000
URL: http://www.securityfocus.com/focus/ids/articles/portscan.html (Sept 18th 2000)

6. Spitzner, Lance, "Passive Fingerprinting" May 2000
URL: http://www.enteract.com/~lspitz/finger.html (Sept 18th 2000)

7. Vision, Max, "Passive Host Fingerprinting" February 2000
URL: http://dev.whitehats.com/papers (Sept 18th 2000)

[1] .Fyodor, "The Art of Port Scanning"
[2] .Fyodor," Remote OS detection via TCP/IP Stack FingerPrinting
[3] Arkin, "Network Scanning Techniques"
[4] Arkin, "Network Scanning Techniques" page 10.
[5] Arkin, page 12.
[6] Spitzner, "Passive Fingerprinting"
[7] Vision, "Passive Host Fingerprinting"
[8] Fyodor," Remote OS detection via TCP/IP Stack FingerPrinting