IP Network Scanning & Reconnaissance

Hacking for Techies™

Technical Primer Booklet

Matta Security Limited

16 – 19 Southampton Place
London WC1A 2AX

+44 (0) 8700 77 11 00

courses@trustmatta.com

http://www.trustmatta.com

Attack & Penetration Group
Matta Security Limited
16 – 19 Southampton Place
London WC1A 2AX

courses@trustmatta.com

+44 (0) 8700 77 11 00

IP Network Scanning & Reconnaissance Technical Primer

Table of Contents

Introduction

Matta is a fiercely independent information risk management company. Since early 2001, the firm has been operating primarily within the UK and Europe – offering bespoke security services, ranging from security assessment, to deployment and integration of authentication, VPN and firewall products.

The latest Matta offering to the already fertile marketplace is an independent, hands-on applied hacking course (with maximum class sizes of 6), broken down into 3 levels –

Level 1 –	Hacking for Newbies™	(1 day course)
Level 2 –	Hacking for Techies™	(2 day course)
Level 3 –	Hacking for Spooks™	(3 day course)

This document is a structured and to-the-point technical primer, comprehensively covering popular (and not so popular) IP network scanning techniques that are adopted by hackers to scan and map networks. Extracts are taken from our Hacking for Techies™ course material, of which more information is available from http://www.trustmatta.com/services/courses.htm. In a bid to increase awareness of information security issues and show just how easy it is to break into computer systems and networks, Matta has decided to openly publish this information and further technical papers into the future. This document is intended as a primer, allowing a structured technical insight into IP level vulnerabilities to be realised.

Matta actively create and present bespoke training programmes to clients with high requirements for information security expertise in-house, allowing them to assess internal network space and other elements themselves. Matta clients that can be mentioned include high street banks, stockbrokers and other financial companies with global footprints. Through it's strong and trusted background, Matta can deliver peace of mind.

For the information of those reading this document and looking to approve it for posting to public forums such as BugTraq, we do not plug our applied hacking courses (or any other commercial offering) at any further stages in this document.

IP Network Scanning & Reconnaissance Techniques

etwork scanning and reconnaissance is the real data gathering exercise of an Internet-based security assessment. The rationale behind network scanning is to achieve the following goals –

- Identify accessible TCP and UDP network services running on the target hosts
- Assess filtering systems between you and the target hosts (firewalls, et al)
- Guess the Operating Systems running by analysing IP responses
- Assess the TCP sequence number predictability of the target hosts for TCP spoofing and sequence prediction attack potential

Upon performing scanning reconnaissance operations, we will have a very good idea of the network topology at hand, and security mechanisms deployed to protect that network environment. The next step from creating this 'network road map' is to glean clear information about the services running, and their versions and enabled options, in order to eventually launch an attack against the target network.

ICMP ping-sweeping and other requests

Upon identifying all the IP addresses and network ranges owned and used by the target organisation, it is sensible to perform an ICMP ping-sweep to identify live accessible hosts in the network ranges.

Nmap is a useful tool for performing ICMP ping-sweeps, as it resolves the names of the hosts and identifies subnet broadcast and network addresses, below is an example of how to perform an nmap ICMP ping-sweep of a network range –

```
$ nmap -sP 192.168.7.1-48

Starting nmap V. 2.54BETA29 by ( www.insecure.org/nmap/ )
Host   (192.168.7.16) seems to be a subnet broadcast address (returned 1 extra
pings).  Skipping host.
Host cube.trustmatta.com (192.168.7.17) appears to be up.
Host onyx.trustmatta.com (192.168.7.18) appears to be up.
Host darkside.trustmatta.com (192.168.7.21) appears to be up.
Host   (192.168.7.31) seems to be a subnet broadcast address (returned 1 extra
pings).  Skipping host.
Host   (192.168.7.32) seems to be a subnet broadcast address (returned 1 extra
pings).  Skipping host.
Host test1.testbed.org (192.168.7.33) appears to be up.
Host dev1.testbed.org (192.168.7.35) appears to be up.
Host pdc.testbed.org (192.168.7.46) appears to be up.
Host   (192.168.7.47) seems to be a subnet broadcast address (returned 1 extra
pings).  Skipping host.
Host   (192.168.7.48) seems to be a subnet broadcast address (returned 3 extra
pings).  Skipping host.
Nmap run completed -- 48 IP addresses (11 hosts up) scanned in 7 seconds
$
```

From the results of the nmap ‘ping sweep’ scan, live hosts responding to ICMP can be identified and subnet information also. The subnet broadcast and network address information is extremely useful, as you may have ping-sweeped the entire Class-C network that you find a target web server on, only to find that the target organisation owns 16 IP addresses of the block. As with the above example, the target servers that we are assessing exist at trustmatta.com, and the testbed.org hosts and network range seems to belong to another organisation entirely.

Certain security-conscious organisations filter ICMP to mission-critical hosts and networks so that ping-sweeping in this fashion is not effective. Domains including microsoft.com and cert.org filter ICMP at their border routers in this way; to identify active hosts, each IP address in the network space has to be port scanned. It should be noted that forcefully scanning hosts in this fashion can be extremely time consuming.

Non-ECHO ICMP queries can be sent to hosts in target network space, including –

ICMP request type	**ICMP packet type information**
TIMESTAMP REQUEST	ICMP type 13 request packet
TIMESTAMP REPLY	ICMP type 14 reply packet
ADDRESS MASK REQUEST	ICMP type 17 request packet
ADDRESS MASK REPLY	ICMP type 18 reply packet

Non-ECHO ICMP queries are sometimes not blocked by firewalls, and can lead to information being leaked that can be abused by determined attackers to learn of your networks and their structure.

- RFC 792 clearly states ICMP ECHO and TIMESTAMP types and their application.
- RFC 950 clearly states ICMP ADDRESS MASK types and their application.
- Other ICMP packet types of interest may be found in RFC’s 1122 and 1812.

Tools such as SING (send ICMP nasty garbage) can be used to send and receive various ICMP messages effectively, available from many Internet sites.

Identifying Active TCP Network Services

To effectively identify active TCP network services running hosts within the target network, we launch a TCP port scan against the target server. There are three distinct groups of TCP port scanning techniques that are adopted by hackers in the wild, which are listed below –

- Open TCP scanning methods
    - Standard connect() scanning
- Stealth TCP scanning methods
    - Half-open SYN flag scanning
    - Inverse TCP flag scanning
    - ACK flag probe scanning
    - TCP fragmentation scanning
- Third-party and spoofed TCP scanning methods
    - FTP bounce scanning
    - Proxy bounce scanning
    - Sniffer-based spoofed scanning
    - IP ID header scanning

Each individual scanning technique can be used to remotely probe the target network, and identify open, closed and filtered ports on the servers and devices within that environment. However, knowing when to launch the correct type scan against a given environment depends completely on the type of network topology deployed, along with any security mechanisms associated with the target hosts and network.

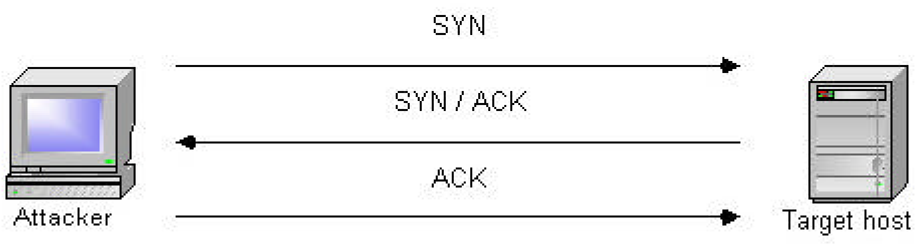
Open scans are a very loud way of scanning a network, and can be detected and logged easily, but they produce highly accurate results of open, closed and filtered network ports. Alternatively, using a stealth scan may bypass security mechanisms (IDS and firewalls) due to the way that obscure packet flags are set, however these packets may be dropped when travelling across networks, leading to false positives being recorded. Each technique has its own pro's and cons, it's simply a case of choosing the most effective scan type to map the target network environment.

Open TCP scanning methods

Standard TCP connect() scanning

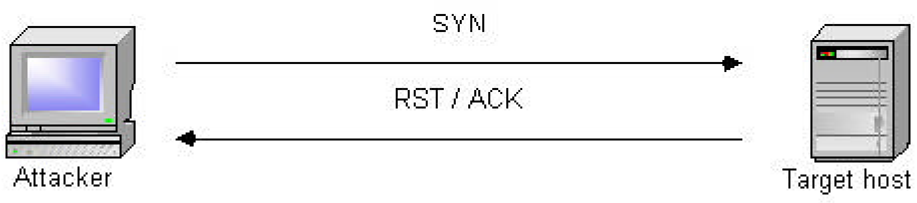
Vanilla TCP port scanning as it is sometimes known, is the most simple type of port scan to launch. There is no stealth whatsoever involved in this form of scanning, as a full TCP/IP connection is attempted to TCP port 1 of the target host, then incrementally through ports 2, 3, 4 and so on. Due to the reliability of TCP/IP as a protocol, vanilla port scanning presents a very accurate way of determining which services are active on the target host.

A vanilla TCP scan result when a port is open –



1. A SYN probe packet is sent to the target host
2. A SYN / ACK packet is received
3. An ACK packet is sent to complete the three-way handshake

A vanilla TCP scan result when a port is closed –



1. A SYN probe packet is sent to the target host
2. An RST / ACK packet is received

Advantages

- Very reliable and accurate way of identifying accessible TCP network services
- Uses unprivileged ports under Unix-based systems, so requires no super-user access

Disadvantages

- Easily logged and detected, a very primitive scanning type

Stealth TCP scanning methods

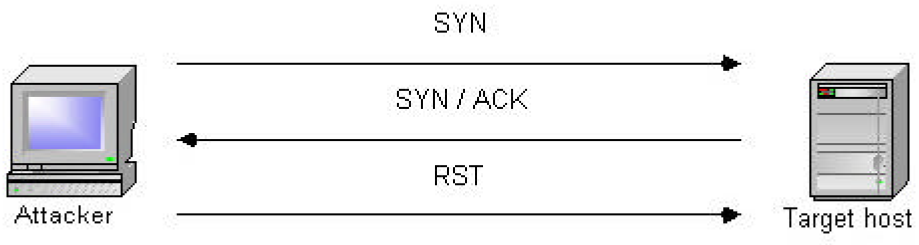
Half-open SYN TCP flag scanning

Stealth scanning in this fashion evades some logging mechanisms because of the fact that a full TCP/IP connection is never established. Usually a three-way handshake is initiated to synchronise a connection between two hosts – the client sends a packet with the SYN flag set to the server, and the server responds with SYN / ACK if the port is open and accepting connections; the client then sends an ACK to complete the handshake.

In the case of half-open SYN port scanning, an RST packet is sent as the third part of the handshake when a port is found to be listening. Sending an RST packet in this way abruptly resets the TCP connection, and due to the fact that you have not completed the three-way handshake, the attempt of the connection is often not logged.

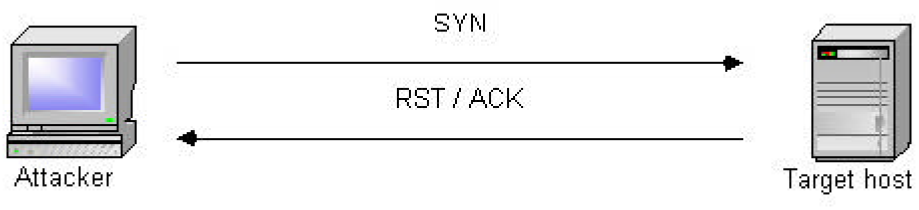
Most Intrusion Detection Systems (IDS) and other security systems including Portsentry can easily detect and prevent stealth half-open scan attempts such as this. It is recommended that in cases where stealth is required, other techniques such as FIN or TTL-based scanning are adopted.

A half-open SYN scan result when a port is open –



1. A SYN probe packet is sent to the target host
2. A SYN / ACK packet is received
3. An RST packet is sent to abruptly reset the connection

A half-open SYN scan result when a port is closed –



1. A SYN probe packet is sent to the target host
2. An RST / ACK packet is received

Advantages

- Fast and reliable way of identifying active TCP network services
- Avoids primitive IDS and logging systems

Disadvantages

- Needs raw access to network sockets, thus requiring super-user privileges
- Network filtering in some cases my block SYN scan attempts

Inverse TCP flag scanning

Filtering and other security systems such as firewalls and IDS can usually pick up on SYN packets being sent to sensitive ports on target hosts. Programs are also available to log half-open SYN flag scan attempts, including synlogger and Courtney. Probe packets with strange TCP flags set can sometimes pass through filters undetected, depending on the security mechanisms deployed.

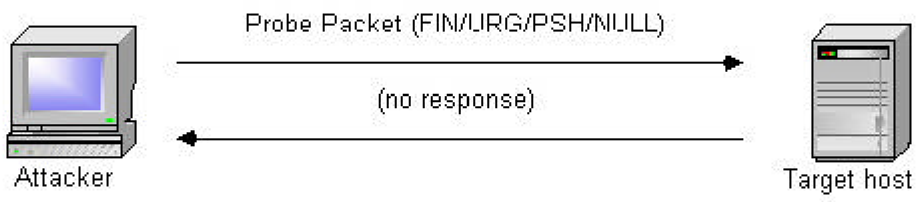
This technique is known as an inverted technique because of the way that we only witness responses sent back from our probes only by closed ports. RFC 793 states that if a port is closed on a host, then an RST / ACK packet should be sent to reset the connection. To take advantage of this feature, we send probe TCP packets with various TCP flags set.

A TCP probe packet is sent to each port of the target host, there are primarily three types of tried and tested probe packet flag configurations that can be used –

- A FIN probe with the FIN TCP flag set
- An XMAS probe with the FIN, URG and PUSH TCP flags set
- A NULL probe with no TCP flags set
- A SYN / ACK probe

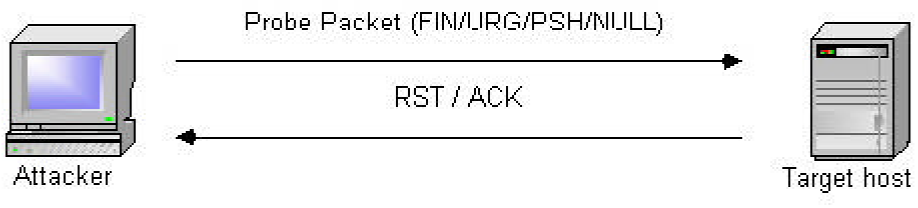
For all closed ports on the target host, packets are received with the RST / ACK flags set. However, operating platforms such as all of those in the Microsoft Windows family, totally disregard the RFC 793 standard, and so no RST / ACK packet is received upon an attempt to connect to a closed port being witnessed. This technique is effective against most Unix-based operating systems.

An inverse TCP scan result when a port is open –



1. A FIN, XMAS or NULL probe packet is sent to the target host
2. If no response is seen, it is assumed that the port is listening

An inverse TCP scan result when a port is closed –



1. A FIN, XMAS or NULL probe packet is sent to the target host
2. An RST / ACK packet is received

Advantages

- Avoids many IDS and logging systems, highly stealthy

Disadvantages

- Needs raw access to network sockets, thus requiring super-user privileges
- Mostly effective against hosts using a BSD-derived TCP/IP stack (not effective against Microsoft Windows hosts in particular)

ACK flag probe scanning

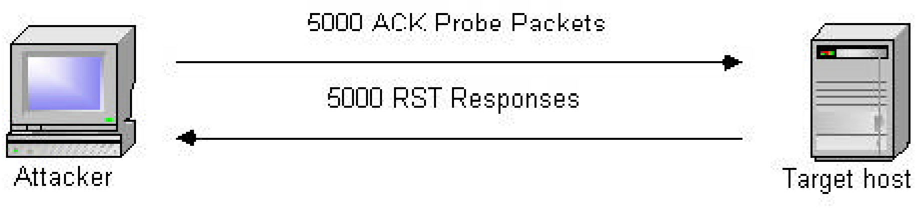
A stealthy technique documented by Uriel Maimon in Phrack issue 49, is that of identifying open TCP ports by sending probe TCP packets with the ACK flag set, and then analysing the header information of the RST packets received from the target host. This technique exploits vulnerabilities within the BSD derived TCP/IP stack, and is therefore only effective against certain operating systems and platforms. There are two main ACK scanning techniques –

- Analysis of the TTL field of received packets
- Analysis of the WINDOW field of received packets

This technique can also be used to check filtering systems and complicated networks to understand the processes that packets go through on the target network. The TTL value can be used as a marker of how many systems the packet has hopped through under certain conditions, and the firewalk filter assessment tool uses a very similar technique.

Analysis of the TTL field of received packets

When analysing the TTL (time to live) field data of received RST packets, we first send thousands of crafted ACK packets to different TCP ports –



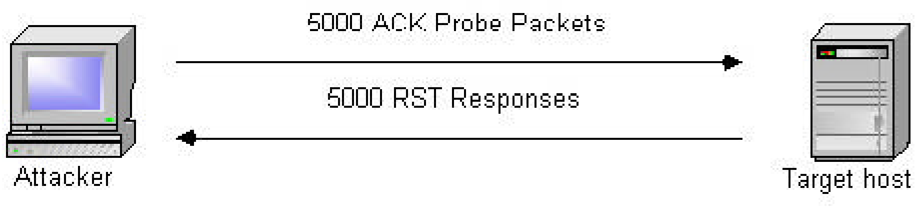
Below is a log of the first 4 RST packets received –

```
1: host 192.168.0.12 port 20: F:RST -> ttl: 70 win: 0
2: host 192.168.0.12 port 21: F:RST -> ttl: 70 win: 0
3: host 192.168.0.12 port 22: F:RST -> ttl: 40 win: 0
4: host 192.168.0.12 port 23: F:RST -> ttl: 70 win: 0
```

By analysing the TTL (time to live) value of each packet, we can easily see that the value returned by port 22 is 40, and the other ports return a value of 70. This would point towards port 22 being open on the target host, as the TTL value returned is smaller than the TTL boundary value of 64.

Analysis of the WINDOW field of received packets

When analysing the WINDOW field data of received RST packets, we first send thousands of crafted ACK packets to different TCP ports –



Below is a log of the first 4 RST packets received –

```
1: host 192.168.0.20 port 20: F:RST -> ttl: 64 win: 0
2: host 192.168.0.20 port 21: F:RST -> ttl: 64 win: 0
3: host 192.168.0.20 port 22: F:RST -> ttl: 64 win: 512
4: host 192.168.0.20 port 23: F:RST -> ttl: 64 win: 0
```

Notice that the TTL for each packet is 64, meaning that TTL analysis of the packets would not be effective in identifying open ports on this host. However through analysing the WINDOW values, we find that the 3rd packet has a non-zero value, indicating an open port.

Advantages

- Avoids IDS detection in most cases, very difficult to identify and log

Disadvantages

- Time consuming to analyse and scan individual host responses in this manner
- Relies on BSD and other TCP/IP stack implementation bugs
- Not effective against later operating platforms

TCP fragmentation scanning

TCP fragmentation scanning is not technically a new scanning type. All we are doing is simply fragmenting our TCP probe packets into thousands of pieces before sending them to the target host. By doing this, we can sometimes bypass simply non-stateful packet filters and even firewalls that are configured for high throughput (and so do not have the time to reassemble and assess packets).

Upon the fragments reaching their target host, they are reassembled and processed at kernel level. Replies will be seen to the probe packets, allowing us to identify open and closed TCP ports on the target host.

The publication of Fragrouter by Dug Song in 1999 as a network intrusion detection system testing tool represented a major milestone in the practical utilisation of IP fragmentation as a means of avoiding detection by network intrusion detection systems.

Fragrouter works by accepting IP packets routed to it by another system, fragmenting those packets according to one of the schemes first described by Ptacek and Newsham in their NIDS evasion paper of 1998, then transmitting the fragmented packets to the target host. The schemes used by Fragrouter to fragment the incoming packets are as follows –

baseline-1	Send the original data in a single TCP data segment.
frag-1	Send the original data in a single TCP data segment, which is broken into 8-byte IP fragments and sent in order.
frag-2	Send the original data in a single TCP data segment, which is broken into 24-byte IP fragments and sent in order.
frag-3	Send the original data in a single TCP data segment which is broken into 8-byte IP fragments, with one of those fragments sent out of order.
frag-4	Send the original data in a single TCP data segment which is broken into 8-byte IP fragments, with the next to last fragment sent twice.
frag-5	Send the original data in a single TCP data segment which is broken into 8-byte IP fragments, sent completely out of order with the next to last fragment sent twice.
frag-6	Send the original data in a single TCP data segment which is broken into 8-byte IP fragments, sending the marked last fragment before any of the others.
frag-7	Send the original data in a single TCP data segment which is broken into 16-byte IP fragments, preceding each fragment with an 8-byte null data fragment that overlaps the latter half of it. This amounts to the forward-overlapping 16-byte fragment rewriting the null data back to the real attack.
tcp-1	Complete a TCP handshake, send fake FIN and RST (with bad checksums) before sending data in ordered 1-byte segments.
tcp-3	Complete a TCP handshake, send data in ordered 1-byte segments, duplicating the next to last segment of each original TCP packet.

tcp-4	Complete a TCP handshake, send data in ordered 1-byte segments, sending an additional 1-byte segment which overlaps the next to last segment of each original TCP packet with a null data payload.
tcp-5	Complete a TCP handshake, send data in ordered 2-byte segments, preceding each segment with a 1-byte null data segment that overlaps the latter half of it. This amounts to the forward-overlapping 2-byte segment rewriting the null data back to the real attack.
tcp-7	Complete a TCP handshake, send data in ordered 1-byte segments interleaved with 1-byte null segments for the same connection but with drastically different sequence numbers.
tcp-8	Complete a TCP handshake, send data in ordered 1-byte segments, with one segment sent out of order.
tcp-9	Complete a TCP handshake, send data in out of order 1-byte segments.
tcb-2	Complete TCP handshake, send data in ordered 1-byte segments interleaved with SYN packets for the same connection parameters.
tcb-3	Do not complete TCP handshake, but send null data in ordered 1-byte segments as if one had occurred. Then, complete a TCP handshake with the same connection parameters, and send the real data in ordered 1-byte segments.
tcbt-1	Complete TCP handshake, shut connection down with a RST, re-connect with drastically different sequence numbers and send data in ordered 1-byte segments.
ins-2	Complete TCP handshake, send data in ordered 1-byte segments but with bad TCP checksums.
ins-3	Complete TCP handshake, send data in ordered 1-byte segments but with no ACK flag set.
misc-1	Thomas Lopatic's Windows NT 4 SP2 IP fragmentation attack of July 1997.
misc-2	John McDonald's Linux IP chains IP fragmentation attack of July 1998.

One of the most interesting facts about Fragrouter is that it is not an attack tool itself, rather it is an enabling technology that allows other attacks to avoid detection by network intrusion detection systems. For example, Fragrouter could be used to obfuscate a phf attack against a web server, a buffer overflow attack against a DNS server, or any number of other attacks. Fragrouter certainly raises the bar for network based intrusion detection systems.

Advantages

- Avoid most IDS systems and network filters deployed

Disadvantages

- Time consuming to deploy fragrouter or a similar system to fragment packets
- Unreliable in some cases due to packet loss and fragmentation
- May cause network and kernel load problems on the target host during reassembly

Third-party and spoofed TCP scanning methods

FTP bounce scanning

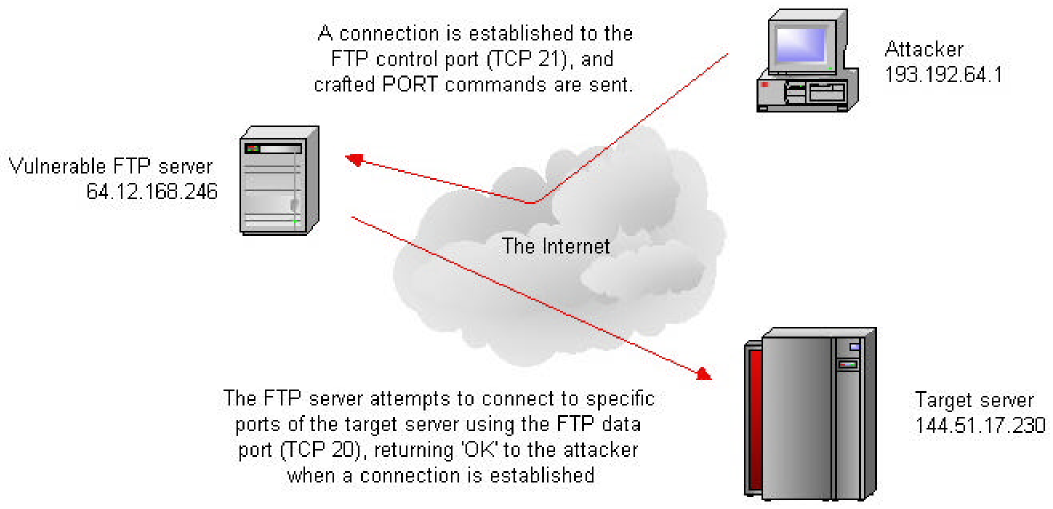
During an FTP bounce port scan, we are abusing vulnerabilities in older FTP network service software, notably the FTP daemons under the following Operating Systems–

- FreeBSD 2.1.7 and earlier
- HP-UX 10.10 and earlier
- SunOS 5.5.1 and earlier
- SunOS 4.1.4 and earlier
- SCO OpenServer 5.0.4 and earlier
- SCO UnixWare 2.1 and earlier
- IBM AIX 4.3 and earlier

- Caldera Linux 1.2 and earlier
- Redhat 4.2 and earlier
- Slackware 3.3 and earlier
- Any Linux distribution running WU-FTP 2.4.2-BETA-16 or earlier

The FTP bounce attack can be used to far more devastating effect if a writable directory exists, as a series of commands or other data can be entered into the file, and then this data can be sent via. the PORT command to a specified port of a target host.

In the case of an FTP bounce port scan being launched, the following occurs –


A connection is established to the FTP control port (TCP 21), and crafted PORT commands are sent.
Attacker
193.192.64.1
Vulnerable FTP server
64.12.168.246
The Internet
The FTP server attempts to connect to specific ports of the target server using the FTP data port (TCP 20), returning 'OK' to the attacker when a connection is established
Target server
144.51.17.230

1. The attacker connects to the FTP control port (TCP port 21) of the vulnerable FTP server that he is going to bounce his attack through, and enters passive mode, forcing the FTP server to send data using DTP (data transfer process) to a specific port of a specific host –

   ```
   QUOTE PASV
   227 Entering Passive Mode (64,12,168,246,56,185).
   ```

2. A PORT command is issued, with an argument being passed to the FTP service, telling it to attempt a connection to a specific TCP port of the target server, in this case port 23 of 144.51.17.230 (0 x 255 = 0 + 23 = 23) –

   ```
   PORT 144,51,17,230,0,23
   200 PORT command successful.
   ```

3. Upon issuing the PORT command, a LIST command is sent. The FTP server then attempts to create a connection with the target host defined in the PORT command issued previously –

   ```
   LIST
   150 Opening ASCII mode data connection for file list
   226 Transfer complete.
   ```

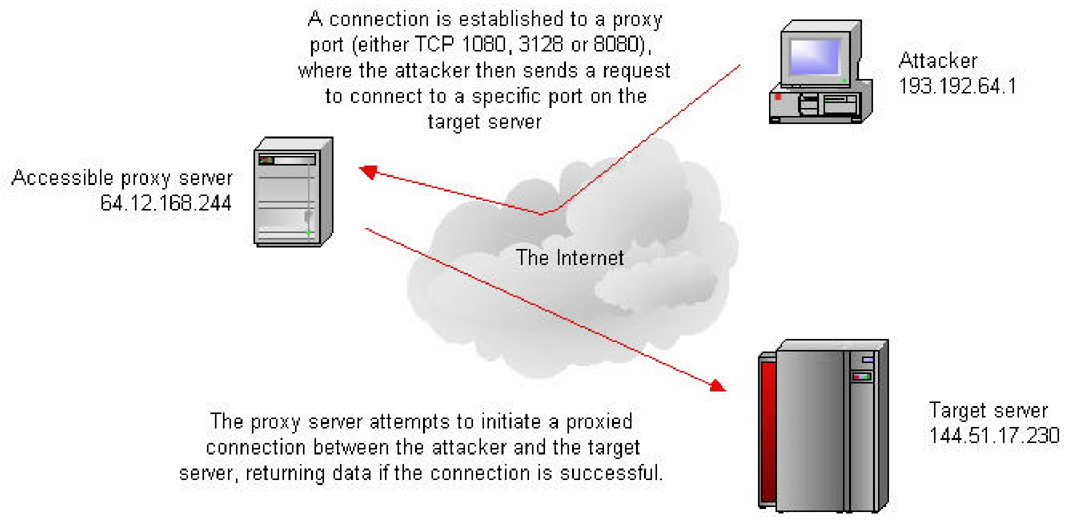
   or..

   ```
   LIST
   425 Can't build data connection: Connection refused
   ```

4. If a return value of 150, followed by 226 is witnessed, then the port on the target server is open. If however a 425 return value is seen, then the connection is refused.

Proxy bounce scanning

Proxy bounce scanning abuses SOCKS-based proxies such as Squid, Microsoft Proxy, or countless others. Below is a diagram –


A connection is established to a proxy port (either TCP 1080, 3128 or 8080), where the attacker then sends a request to connect to a specific port on the target server
Attacker
193.192.64.1
Accessible proxy server
64.12.168.244
The Internet
Target server
144.51.17.230
The proxy server attempts to initiate a proxied connection between the attacker and the target server, returning data if the connection is successful.

Proxy scanning can be slow and cumbersome in cases, depending on the speed of your Internet connection and that of the proxy server which you are abusing. Websites such as CyberArmy have frequently updated listings of open web proxies running on TCP ports 1080, 3128 and 8080, a comprehensive list of which is available from http://www.cyberarmy.com.

Sniffer-based spoofed TCP scanning

A new breed of publicly available scanner is spoofscan.c by jsbach, which is available from Packet Storm (currently at www.packetstormsecurity.org), under the UNIX -> Scanners. section Spoofscan takes advantage of a fundamental vulnerability in shared network segments which allows such spoofing and sniffing attacks to take place.

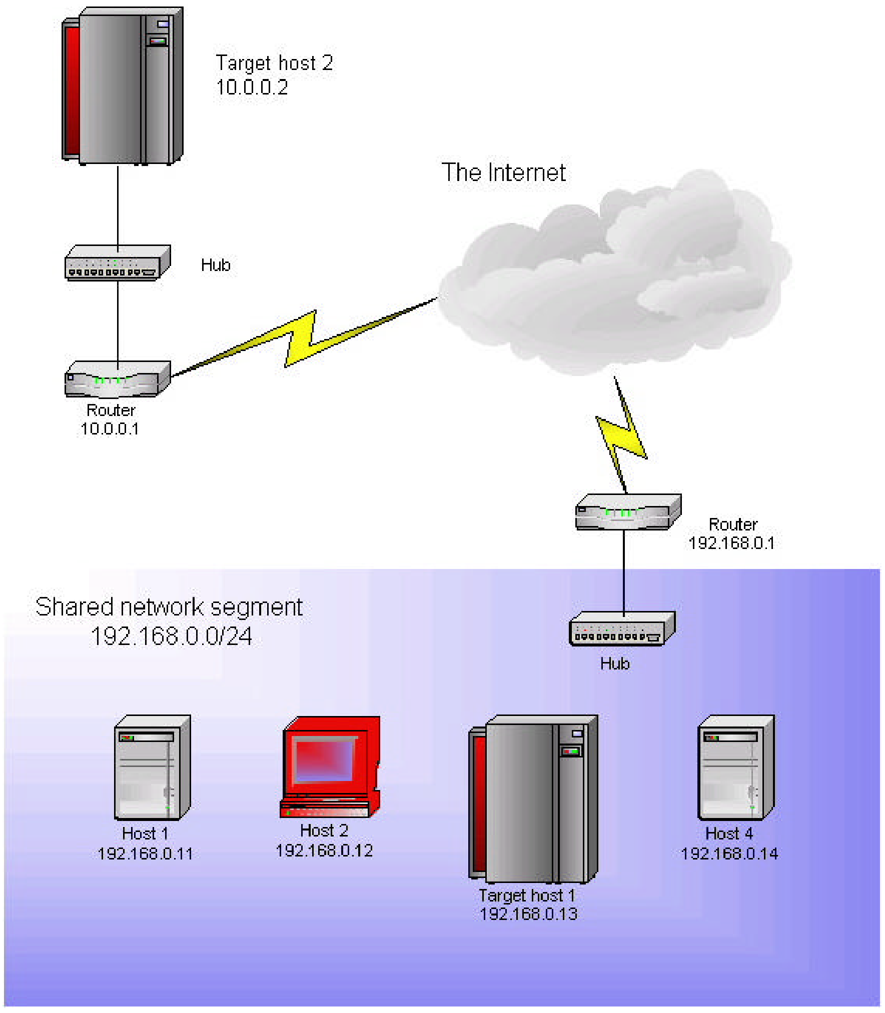
Spoofscan works by sending out spoofed TCP/IP packets with a different source IP address to your own, and then sniffs the responses as they come back to your network segment.

For this to work however, you have to either be on –

- The same shared network segment as the host you want to fake the scans from
- The same shared network segment as the target host that you want to scan
- Somewhere in between, on the same network segment as the router or gateway host which connects the target host directly or in-directly to the Internet

It also has a distinct benefit when evading pro-active IDS systems which may block scans from IP addresses that have been logged. If you have super-user access to a host on a shared class-c network segment of 254 IP addresses, you can spoof your port scan as originating from each and every routable IP address in the address space. There are various other scenarios when using spoofed port scans in this way.

The three basic scenarios are explained below with the following diagram –



In the diagram, we have root access to host 2 and jsbach's spoofscan utility installed. Due to the fact that spoofscan sends out spoofed probe packets and then sniffs the responses using the shared network segment, we can spoof port scans from any host in the 192.168.0.* address space launched against the target host 2 across the Internet at 10.0.0.1.

In the same way, we could spoof a port scan launched against target host 1 from any IP address, including 1.2.3.4 and 1.3.3.7. This technique can be used as a nifty DoS if portsentry or a pro-active security system has been deployed and is configured incorrectly. We could systematically spoof port scans from trusted and depended hosts, which would then be written into the hosts.deny file on target host 1, and not be able to connect to the server later.

A slightly more theoretical way of performing spoofed port scans in this fashion would be to gain access to a host that lies on a static route between the 192.168.0.0 and 10.0.0.0 networks, there are various possibilities depending on networking conditions in place.

IP ID header TCP scanning

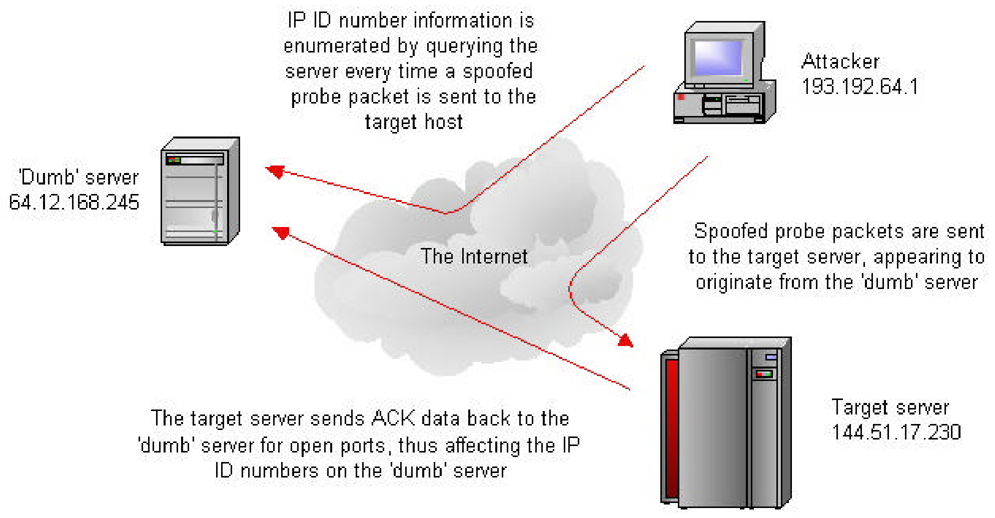
ID header scanning is an obscure scanning technique, which involves abusing implementation peculiarities within the TCP/IP stack of most operating systems. TCP port scanning in this fashion is not easily done, and is a complicated, unreliable and time-consuming way of identifying open TCP ports on target hosts. However it is an extremely stealthy technique, which will defeat IDS or other security systems deployed on the target network environment from identifying the true source of your probing.

Three hosts are involved –

- Your host, from which the scan is launched.
- The target host, which you will be scanning.
- A ‘dumb’ host, an Internet-based server which is queried in-line with spoofed port scanning against the target host, to identify open ports.

An example of a dumb host would be a bastion host, or a server that receives little or no network-based traffic at all. Locating a dumb host requires much network probing and scanning in itself, and is probably more trouble than it is really worth. In the technique documented here, we need to be able to send ICMP packets to the dumb host.

Upon finding such a dumb host, the following diagram documents the technique of identifying open ports on the target server –


IP ID number information is enumerated by querying the server every time a spoofed probe packet is sent to the target host
Attacker
193.192.64.1
'Dumb' server
64.12.168.245
The Internet
Spoofed probe packets are sent to the target server, appearing to originate from the 'dumb' server
The target server sends ACK data back to the 'dumb' server for open ports, thus affecting the IP ID numbers on the 'dumb' server
Target server
144.51.17.230

Involved in this scenario are three hosts –

A – The attacker’s host
B – The ‘dumb’ server
C – The target server

The attacker first uses hping to continuously poll the dumb server and retrieve IP ID header information, sending data from host A to B –

```
$ hping dumb.example.com -r
HPING dumb.example.com (eth0 64.12.168.245): no flags are set, 40 data bytes
60 bytes from 64.12.168.245: flags=RA seq=0 ttl=64 id=41660 win=0 time=1.2 ms
60 bytes from 64.12.168.245: flags=RA seq=1 ttl=64 id=+1 win=0 time=75 ms
60 bytes from 64.12.168.245: flags=RA seq=2 ttl=64 id=+1 win=0 time=91 ms
60 bytes from 64.12.168.245: flags=RA seq=3 ttl=64 id=+1 win=0 time=90 ms
60 bytes from 64.12.168.245: flags=RA seq=4 ttl=64 id=+1 win=0 time=91 ms
60 bytes from 64.12.168.245: flags=RA seq=5 ttl=64 id=+1 win=0 time=87 ms
```

As long as a steady incremental response is gauged, this host can be used to launch an IP ID attack against host C – the target. In a nutshell, we are using the dummy host B as an indicator as to if the port is open on target host C, or not.

The next step is for the attacker at host A to send a series of spoofed SYN packets to a the port that he wishes to scan on host C, pretending to have come from the 'dumb' server B –

```
$ hping 144.51.17.230 -a 64.12.168.245 –S
```

If the port on target host C is listening and accepts the connection, the following will occur –

1. A SYN / ACK response packet will be sent from target host C to dumb host B.

2. Dumb host B will know nothing of this connection attempt, and so a RST packet will be sent from host B to C.

Upon an RST packet being sent from the dumb host B to the target host C, the IP ID header counter is affected and incremented on dumb host B for each RST packet sent. If we send 3 spoofed packets to target host C, we would expect to see the following change in the IP ID header numbers –

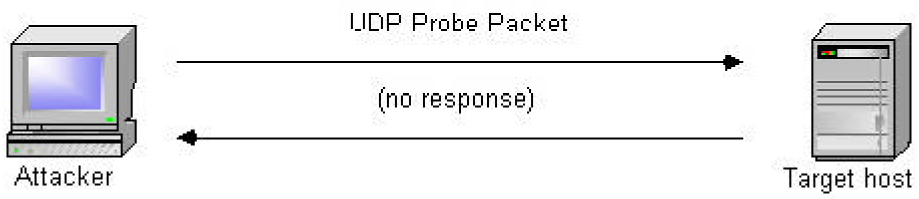
```
60 bytes from 64.12.168.245: flags=RA seq=17 ttl=64 id=+1 win=0 time=96 ms
60 bytes from 64.12.168.245: flags=RA seq=18 ttl=64 id=+1 win=0 time=80 ms
60 bytes from 64.12.168.245: flags=RA seq=19 ttl=64 id=+2 win=0 time=83 ms
60 bytes from 64.12.168.245: flags=RA seq=20 ttl=64 id=+3 win=0 time=94 ms
60 bytes from 64.12.168.245: flags=RA seq=21 ttl=64 id=+1 win=0 time=92 ms
60 bytes from 64.12.168.245: flags=RA seq=22 ttl=64 id=+1 win=0 time=82 ms
```

Identifying Active UDP Network Services

UDP (User Datagram Protocol) is a connectionless protocol, as datagrams are sent between hosts and may be discarded before reaching their targets. UDP as a protocol is useful when implementation of TCP would be too complex, too slow, or just unnecessary.

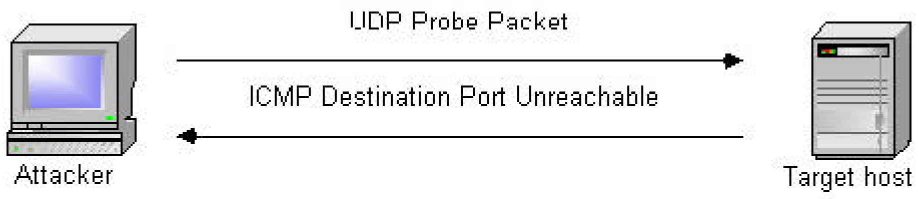
Due to the protocol itself and the way it is implemented, it is often difficult to identify open UDP ports on most networks that we will encounter. The most common way currently of identifying open UDP ports is to adopt an inverted technique; involving sending a UDP probe packet to a specific port, and listening for an 'ICMP Destination Port Unreachable' message, if such a message is received, then the target port is closed. Through a process of deduction, we can identify listening ports, by eliminating those that return an ICMP unreachable message. This method is moderately effective in most cases, but depends on UDP and ICMP packet filtering between us and the target host. Below are two examples –

An inverse UDP scan result when a port is open –



1. A UDP probe packet is sent to the target host
2. If no response is seen, it is assumed that the port is listening

An inverse UDP scan result when a port is closed –



1. A UDP probe packet is sent to the target host
2. An 'ICMP Destination Port Unreachable' message is received

Operating System Guessing Techniques

Various Operating Systems have their own interpretations of IP level standards when receiving certain types of packets and responding to them. Often, by analysing responses from Internet-based hosts carefully, attackers can guess the operating platform of the target host via. IP fingerprinting, commonly using TCP, ICMP and UDP techniques –

- TCP FIN probes and bogus flag probes
- TCP sequence number sampling
- TCP WINDOW and ACK value sampling
- ICMP message quoting
- ICMP ECHO integrity
- Responses to IP fragmentation
- IP TOS (type of service) sampling

Originally, tools such as Cheops and Queso were specifically developed to guess target system operating platforms. The first publicly available tool to perform this was sirc3, which simply detected the difference between BSD-derived TCP stacks, Windows TCP stacks and Linux TCP stacks.

Nowadays, comprehensive port scanning utilities such as nmap are highly effective at identifying operating platforms in most cases. There are still idiosyncrasies that you must get used to when using nmap however, such as nmap thinking that certain firewalls are AIX 4.x servers!

Network Service Specific Reconnaissance Techniques

Banner Grabbing

The following TCP ports can usually be queried in plaintext to give useful banner information, usually documenting service information and enabled options –

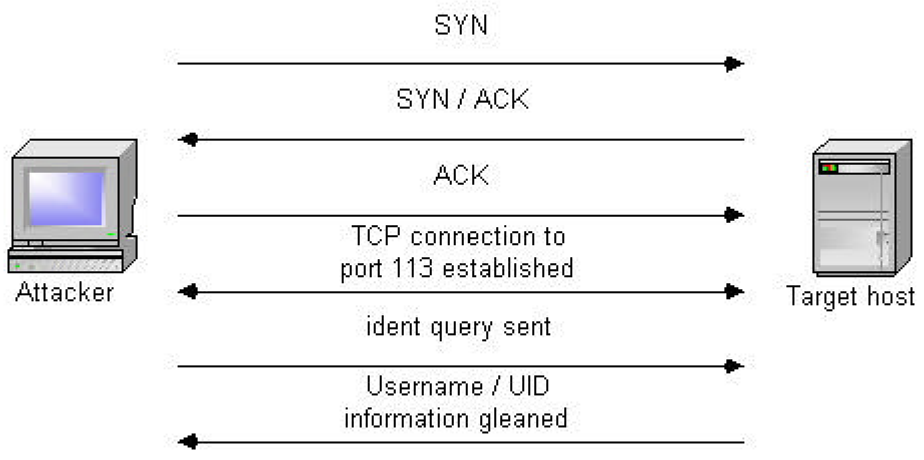
Port	**Service**	**Example Banner**
21	FTP	onyx.trustmatta.com FTP server (SunOS 5.6) ready.
22	SSH	SSH-1.99-OpenSSH_3.0p1
23	Telnet	Unix System V Release 4.0 (mail.trustmatta.com)
25	SMTP	Sendmail 8.8.8/8.8.8+Sun ESMTP
80	HTTP	Apache 1.3.9 (Win32)
109	POP2	POP2 mail.trustmatta.com v4.46 server ready
110	POP3	Microsoft Exchange POP3 server version 5.5.2653.23 ready
143	IMAP2	IMAP4rev1 mail.trustmatta.com v11.241 server ready
3128	Squid	Squid/2.3.STABLE1

Ident-based Port Querying

Instead of listing ident-based port querying as a network scanning type, it is more or a service specific reconnaissance technique. Upon finding that the ident service is accessible, we can issue ident queries as to who owns the processes running on accessible TCP ports. Ident-based port querying is an expansion of standard connect() TCP port scanning.

The ident service is traditionally used on Unix-based systems to identify users when they are connecting to foreign systems and give a level of username-based authentication and auditing. The ident service runs on TCP port 113 by default, and is queried with a TCP port pairing, upon receiving this query string, the ident service returns the username or UID of the owner of the open port.

A reverse ident scan result when a port is open –


SYN
SYN / ACK
ACK
TCP connection to port 113 established
ident query sent
Username / UID information gleaned
Attacker
Target host

1. A SYN probe packet is sent to the target host
2. A SYN / ACK packet is received
3. An ACK packet is sent to complete the three-way handshake

4. Open identifying an open port, an ident connection is established
5. An ident query of the open port is sent
6. Username or UID information is gleaned

Network Scanning Tools Listing

Nmap	Nmap is a Unix-based TCP and UDP port scanning utility, Comprehensively covering all of the major port scan types, even bounce attacks, OS guessing and network querying.
Spoofscan.c	Spoofing / sniffing-based TCP portscanner by jsbach, Nmap does not perform this scanning type!
Fragrouter	IDS and logging evasion system, fragments packets in many different modes.
IDES.c	Another IDS evasion system, adds various RST and other packets to data streams in order to confuse IDS mechanisms.
Firewalk	ACL assessment utility with limited mileage, uses TTL-based analysis of packets to identify filtered and non-filtered ports.
HPING	Utility to send and analyse specific TCP packet types.
SING	Utility to send and analyse specific ICMP packet types.

Closing Comments

At the time of publishing this primer booklet presenting a structured insight into IP Network Scanning and Reconnaissance, the following Matta white papers are available publicly, giving clear technical and conceptual insight to other issues at hand –

- An Introduction to Internet Attack & Penetration
- Using DNS to Effectively Map Networks
- Denial of Service Technical Primer

Available from the Matta website at http://www.trustmatta.com, along with other information and security white papers which may be of interest. Into the future, technical information regarding specific attack types and hacker strategies will become available, so keep posted.